;login:

The USENIX Association Newsletter

Volume 10, Number 3 **August 1985**

CONTENTS

Call for Memorabilia for the 10th Anniversary Issue of ;login:

The next issue of *;login:* will be a special one to commemorate the 10th Anniversary of UNIX user group meetings. We need articles, anecdotes, recollections, historical documents, pictures, meeting announcements and programs, records of early meetings (especially those before 1980), records of awards, audio tapes – anything that might be of interest and/or help document the record.

Articles and letters to the editor with information, anecdotes, and histories of key milestones and personalities in the UNIX and USENIX world are particularly welcome.

Please send your contributions to Lou Katz at:

ucbvax!lou

or to the Association office.

When sending hard copy, please send originals if possible as they will reproduce better. Original materials will be returned unless you give permission for them to be kept in the Association's archives.

The closing date for submissions for the next issue of *;login:* is August 23, 1985

NOTICE

;login: is the official newsletter of the USENIX Association, and is sent free of charge to all members of the Association.

The USENIX Association is an organization of AT&T licensees, sub-licensees, and other persons formed for the purpose of exchanging information and ideas about UNIX† and UNIX-like operating systems and the C programming language. It is a non-profit corporation incorporated under the laws of the State of Delaware. The officers of the Association are:

President	Alan G. Nemeth
Vice-President	Deborah K. Scherrer
Secretary	Lewis A. Law
Treasurer	Waldo M. Wedel
Directors	Thomas Ferrin
	Steve C. Johnson
	Lou Katz
	Michael D. Tilson
Executive Director	James E. Ferguson

The editorial staff of *;login:* is:

Publisher	James E. Ferguson usenix!jim
Technical Editor	Brian Redman {bellcore,usenix}!ber
Copy Editor	Michelle Peetz {masscomp,usenix}!mmp
Managing Editor	Tom Strong usenix!tom
Editorial Director	Lou Katz {ucbvax,usenix}!lou

Member Services

Member services are provided through the Association office. Membership information can be obtained from the office:

USENIX Association
P.O. Box 7
El Cerrito, CA 94530
(415) 528-8649
{ucbvax,decvax}!usenix!office

Contributions Solicited

Members of the UNIX community are heartily encouraged to contribute articles and suggestions for *;login:*. Your contributions may be sent to the editors electronically at the addresses above or through the U.S. mail to the Association office. The USENIX Association reserves the right to edit submitted material.

;login: is produced on UNIX systems using *troff* and a variation of the –me macros. We appreciate receiving your contributions in *n/troff* input format, using any macro package. If you contribute hardcopy articles please leave left and right margins of 1″ and a top margin of 1½″ and a bottom margin of 1¼″. Hardcopy output from a line printer or most dot-matrix printers is not reproducible.

Acknowledgments

The Association uses a VAX‡ 11/730 donated by the Digital Equipment Corporation for support of office and membership functions, preparation of *;login:*, and other association activities. It runs 4.2BSD, which was contributed, installed, and is maintained by mt Xinu. The VAX uses a sixteen line VMZ-32 terminal multiplexor donated by Able Computer of Irvine, California.

Connected to the VAX is a QMS Lasergrafix* 800 Printer System donated by Quality Micro Systems of Mobile, Alabama. It is used for general printing and draft production of *;login:* with *ditroff* software provided by mt Xinu.

†UNIX is a trademark of AT&T Bell Laboratories.

‡VAX is a trademark of Digital Equipment Corporation.
*Lasergrafix is a trademark of Quality Micro Systems.

To the Editors:

Some Comments on the Portland Meeting

By far the neatest session of the conference was at the end, featuring Skudlarek (plus Lyon) on botches in chip design, Pike (plus Weinberger) on botches in name syntax design, and Redman (plus Cross) on fun with phones. The first and third were especially well presented and surpassed their printed versions; the second was misunderstood by those for whom it was intended, and thus may stand as an entertaining failure.

Also memorable was the mail session in which Presotto, Ostby, and Allman feasted on the corpse of Allman, who seemed revived by the experience.

What I'll remember longer is the man-made fireworks at the banquet, and longer yet, the results of the natural fireworks around Mount St. Helens.

Dennis Ritchie

Future Meetings of the USENIX Association, EUUG, and AUUG

August 26-27, 1985: Brisbane, Queensland, Australia

The Winter National Conference of the Australian UNIX systems Users' Group (AUUG) will be held in Brisbane on August 26-27, 1985, at Queensland University. Keynote speaker for the meeting is Stuart Feldman, author of *make* and the first *f77* compiler. For further information, call Tim Roper on +61 7 377 2875 or send mail to

auug%uqcspe.oz@seismo.arpa or seismo!munnari!uqcspe.oz!auug

September 10-13, 1985: Copenhagen, Denmark

The next meeting of the European UNIX system User Group (EUUG) will be held September 10-13, 1985, at the Bella Center in Copenhagen. Accommodations are in downtown Copenhagen in hotels situated around the famous Tivoli gardens. There will be tutorials, industry sessions, technical sessions, and an exhibition. For further information, registration forms, etc., contact:

DIS Congress Service
Attn: Birthe Kanstrup
48, Linde Alle
DK-2720 Vanløse/Copenhagen, Denmark

Tel: +45 1 712244
Telex: 15476 dis dk

January 15-17, 1986: Denver, Colorado

The USENIX Winter '86 Conference will be held in Denver on January 15-17, 1986, at the Marriott Hotel. Emphasis will be on a series of intensive workshop seminars; there will also be tutorials. Please see the calls for participation and papers elsewhere in this issue of *;login:*. To avoid conflicting with the 1986 UniForum in Anaheim, there will be no formal vendor exhibition; however, vendors are invited to have hospitality suites.

June 10-13, 1986: Atlanta, Georgia

The USENIX Summer '86 Conference will be held in Atlanta on June 10-13, 1986. There will be a conference, tutorials, and vendor exhibits.

June 9-12, 1987: Phoenix, Arizona

The USENIX Summer '87 Conference will be held in Phoenix on June 9-12, 1987. There will be a conference, tutorials, and vendor exhibits.

CALL FOR PAPERS

USENIX Computer Graphics Workshop

December 12-13, 1985
DoubleTree Hotel
Monterey, California

USENIX is sponsoring the second annual workshop on current and future developments in computer graphics in the UNIX environment or with UNIX tools and/or philosophy.

The workshop will be structured to facilitate in-depth discussions of technical issues, and will have presentations in a number of formats, with ample time for questions and responses. There will be a computer graphics film and video presentation.

Presentations will be 20-40 minutes in length, and speakers are encouraged to include visual examples and illustrations in the media of their choice.

Deadline for submissions is September 15, 1985. Submissions must consist of a camera-ready or electronic copy of a paper to be included in the proceedings as well as a copy of any visual materials to be included in the videotape proceedings. Color slides and other visuals will be transferred to videotape. All original materials will be returned.

Send all materials to:

Reidar J. Bornholdt
Room 7-444
Columbia University
College of Physicians & Surgeons
630 West 168 Street
New York, NY 10032

{harpo|cmcl2}!cucard!reidar or {ucbvax|decvax}!usenix!reidar

Program Committee:

Reidar Bornholdt, Columbia University, Chair
Lou Katz, Metron Computerware
Peter Langston, Bell Communications Research
Tom Duff, Bell Laboratories

Suggested topics include:

Animation
Text processing and phototypesetting
Windows
Management of picture information storage
Rendering
Strange input/output hardware

Domain Addressing in ACSnet

Piers Lauder
Sydney University

ABSTRACT

A form of hierarchical addressing employing the concept of *domain* has been introduced to the Australian Computer Science Network. Domains can be equivalent to geographical addresses, and allow a uniform presentation of addresses that are universally valid.

Introduction

The concept of a *domain* as part of a network address is presently generating considerable discussion on USENET. Some of the discussion seems to revolve around exactly which bureaucratic entities will be institutionalised by being allowed their own domain name. It appears that this will then allow them control over the routing of network messages.

There is another point of view.

There is no reason why domains should be any different from the geographical entities used by postal agencies around the world to deliver mail. That is, domains can be hierarchical, map to geographically distinct areas, and impose no control over the routing of messages. Why not anticipate the future, when there will surely be no difference for users, whether they are addressing a postal item, or a computer message?

This paper describes the introduction of domains into the Australian Computer Science Network (a.k.a. ACSnet). Previously, this was a dynamic routing network requiring only a user name and machine name to completely deliver a message.

Domains and Network Addresses

Computer message addresses are changing. As computer networks are interconnecting to form a world-wide net, there is a need for computer message addresses as universally valid and as easy to use as postal addresses or telephone numbers.

Until now, those of us using ACSnet have been living in a simple community, as though everyone lived in the same street and it was only necessary to know people's names and house numbers to identify them uniquely. Now a city has grown up around us, with links to other cities and countries, and street addresses are no longer unique.

The concept of a *domain* has been introduced[1] to describe the arrangement of addresses within computer networks. A domain is a grouping of computers to support a common purpose, such as all those computers running a particular type of network software, or all the computers in a university campus, or all the computers in a particular country. Domains enable us to generate a computer network address that can not only be used to deliver a message to a local destination, but will also deliver a message to the same destination from anywhere in the world.

For example, the following table shows a traditional postal address with the components distinguished by separate lines, together with the equivalent components from a network address:

[1] ARPA RFC 882

Piers Lauder	piers
Computer Science	cs
Sydney University	su
Australia	oz

When presented to the network, the address would appear as

piers.cs.su.oz

(the line separators have been replaced by periods.)

Previously, the network address for this destination would have been

piers:basser

which has only two components and is probably easier to remember, but was only valid in Australia on ACSnet sites.

The **:basser** is a node identifier—the name of a particular computer—and it is not something everyone should be required to remember.

There are potentially thousands of network nodes in Australia alone, and for this address form to work, there must be some data-base that knows how to deliver a message to every one of them.

By contrast, for the domain addresses to work anywhere in the world, a local data-base need only retain knowledge of the routes to those major domains outside, and to each of the domains contained within, its own domain. An analogy is telephone switching, where each exchange need only remember its own local numbers, and know how to switch calls out to other area codes.

We have sacrificed some "user friendliness" to reduce a computing task, but consider the problem of communicating with other continents. Here is the address that must be used to reach a colleague in California at UC San Diego Computer Science from anywhere in Australia:

decvax!ucbvax!sdcsvax!sdamos!john:mulga

Here is a more arcane example:

decvax!ucbvax!@SU-SCORE.ARPA:mike%unc.csnet@csnet-relay:mulga

This routes first via ACSnet to **mulga**, then to **decvax** and **ucbvax** by **uucp**, then by CSNET to **csnet-relay**, then to **unc.csnet** using CSNET, which then sends to SU-SCORE via ARPANET for delivery to user **mike**. *Caveat sender.*

The domain version of this might be

mike.SU-SCORE.ARPA.CSNET.UUCP

but this just reproduces the problems of explicit routing, using domains instead of nodes. What would be best of all is the domain version of the postal address:

mike.cs.stanford.usa

Although addresses have become slightly more complex for communicating within Australia, they will become much less complex for the general case. And of course there are abbreviated forms for addresses within a local domain. In the same way that a letter addressed to someone in Sydney from someone in Melbourne need never specify "Australia," a message from Melbourne to Sydney need not specify '.oz'. So, for example, here are the addresses needed to reach the author from various places:

Message source	Address
Computer Science, Sydney University	piers
Psychology Department, Sydney University	piers.cs
Computer Science, Melbourne University	piers.cs.su
Bell Laboratories, NJ, USA	piers.cs.su.oz

CALL FOR PARTICIPATION

USENIX Winter '86 Conference

January 15-17, 1986
Marriott Hotel
Denver, Colorado

The Winter '86 Conference will consist of workshop-oriented technical sessions as well tutorial classes in three topic areas:

- Window Environments and UNIX – Wednesday, January 15
- UNIX on Big Iron – Thursday, January 16
- Ada and the UNIX System – Friday, January 17

Each day of the technical sessions will be devoted to only one topic. Attendees may attend one, two, or three days. For each topic area, there will be related tutorials on adjacent days, concurrent with the other technical sessions.

There will also be several tutorials of general interest. Tutorial topics may include: IPC, streams, and sockets; SNA networking; performance analysis and benchmarking; Ada language; Ada programming; etc.

We need:

- Program committee members
- Sessions chairpersons
- Tutorial speakers

We are also soliciting ideas for specific session topics and other tutorial topics. If you are interested, please contact:

Jim Ferguson
Executive Director
USENIX Association
P.O. Box 7
El Cerrito, CA 94530

415-528-8649
{decvax|ucbvax}!usenix!jim

Call for Papers: Window Environments and UNIX[tm]

USENIX Winter Conference
Denver, Colorado
January 15, 1986

This meeting will explore the design and integration of UNIX-based window systems and their applications. We solicit papers for presentation; topics might include but are not limited to:

- hardware and software architectures
- window system – operating system interface issues
- window-based programming environments
- color: use, implementation, integration
- user interface design
- integration of windows and existing UNIX tools

Papers or extended abstracts must be submitted by September 15. Authors will be notified of acceptance by October 15. Camera-ready copy of accepted papers must be received by December 1. Papers should be 8 to 12 pages long. Presentations will be limited to 30 minutes. Proceedings will be distributed at the conference.

Send submissions and queries by electronic mail to **ucbvax!windex**, or by US mail to:

Sam Leffler
P.O. Box 2009
San Rafael, CA 94912
415-499-0239

CALL FOR PARTICIPANTS

UNIX† on Big Iron

USENIX Winter Conference
Denver, Colorado
January 16, 1986

On January 16, 1986, USENIX will devote one day of its annual Winter Conference to exploring issues raised by the implementation and operation of UNIX on very large or powerful mainframes. We solicit papers for presentation during this day.

The deadline for submissions is September 15, 1985. Camera-ready copy will be required by December 1, 1985. Proceedings will be distributed at the conference.

For further information, please contact:

Peter Capek, Chair
IBM Research
P.O. Box 21
Yorktown Heights, NY 10598
914-945-1250
capek.yktvmv.ibm@udel-relay

†UNIX is a trademark of AT&T Bell Laboratories.

CALL FOR PAPERS

Ada† and the UNIX‡ System

USENIX Winter Conference
Denver, Colorado
January 17, 1986

On January 17, 1986, USENIX will devote one day of its annual Winter Conference to an examination of the Ada language and its relation to the UNIX system. We solicit papers for presentation during this day. Topics might include, but are not limited to:

- hosting or targeting Ada compilers on UNIX systems;
- Ada software development environments on UNIX systems;
- UNIX tools and their usefulness with Ada;
- large project development and UNIX systems;
- standardization of Ada runtime system or environments and UNIX standards;
- experience using Ada on UNIX systems; and
- . . . (let your imagination be your guide) . . .

Papers and extended abstracts are acceptable; they should be 8 to 12 pages long. The papers will be reviewed by the program committee. Submission date is September 15, 1985. Authors will be notified of acceptance by October 15, 1985, and camera-ready copy will be required by December 1, 1985. Proceedings will be distributed at the conference.

Please send five copies of papers to be reviewed to:

Charles Wetherell
AT&T Information Systems
Room F-326
190 River Road
Summit, NJ 07901
201-522-6365
attunix!cw

†Ada is a registered trademark of the U.S. Government – Ada Joint Program Office.
‡UNIX is a trademark of AT&T Bell Laboratories.

Using fsck

A Guide to the UNIX† File System Check Program

Michael S. Saxon
SGS Semiconductor Pte. Ltd.
Singapore

1. Introduction

The UNIX File System Check program, **fsck**, checks the disk to make sure a proper environment exists in which to create and maintain files. This is called *consistency*. If the file system is inconsistent, corrective action must be taken before the file system can be used.

This article is basically a re-write of the article "FSCK – The UNIX/TS File System Check Program" by T. J. Kowalski of Bell Laboratories. While Kowalski's article provides a complete portrait of **fsck**, it is not written in a fashion which is understandable by the non-technical personnel who are now called upon to use this life-saving program.

In some instances, general information not provided in Kowalski's article has been added to make some portions of **fsck** more clear. In other cases, some more esoteric items have been left out.

For UNIX purists, the original article remains the "Bible" on **fsck**. This article attempts to fill the gap between the dangerous ignorance with which **fsck** is often used today and the gospel version which precedes it.

[Editor's Note: Some of the details described in this article do not apply to the 4.2BSD or 4.3BSD versions of the UNIX filesystem. Readers are referred to "Fsck – The UNIX File System Check Program" by Marshall Kirk McKusick, revised July 28, 1983. This paper is reprinted in Volume 2 of the 4.2BSD UNIX System Manager's Manual. The prudent 4.2/4.3 UNIX user is advised to supplement what follows with the specific details found in the above reference.]

2. Using fsck

fsck should be used whenever a computer using the UNIX operating system is started up. UNIX should maintain the integrity of the file system on its own while it is running.

fsck should always be run in the UNIX single-user mode. This insures that the operator has complete control over the disk system while **fsck** is running. A file system which is not doing anything is called a *quiescent* file system.

When **fsck** checks a file system, it does so in multiple passes. It is because of the multi-pass nature of **fsck** that the file system must be quiescent when **fsck** is run. If this is not the case, the running computer system could undo changes being made by **fsck** or could crash as a result of changes made by **fsck**.

Lastly, as certain information concerning the file system is kept in memory at all times (the *super-block* discussed in Section 4) it may be necessary to reset the computer immediately after running **fsck**. This is because if a *sync* operation is performed, the old version of the super-block will be re-written to the disk, thus overwriting the newly corrected version.

Please note that when a UNIX system is in multi-user mode, the sync operation is usually performed every 30 seconds by the *update* process. This means that changes made to a system in multi-user mode by **fsck** could be undone by the update process. This is another reason why **fsck** should always be run in single-user mode.

---

†UNIX is a trademark of AT&T Bell Laboratories.

If possible, **fsck** should be run on unmounted file systems to prevent these problems from occurring. This is because only the super-blocks of mounted file systems are updated by UNIX while it is running. If it is impossible to unmount the file system in question, the system **MUST** be reset after **fsck** is run if any changes were made. **fsck** will warn you that this is necessary with the message "BOOT UNIX (NO SYNC!)."

If **fsck** finds an inconsistency in a file system, it reports it immediately to the operator and asks the operator whether or not to take action. The reason why **fsck** does not take any action on its own is that there may be multiple courses of action available. Also, the operator may be working on a known bad file system and may not wish to take any steps which would do further damage while in the process of trying to save files.

fsck must create a number of large tables in order to check a file system. Usually these tables are created in the main memory. If there is not enough memory space available for all the tables, **fsck** will prompt the operator for a place to create a scratch file. Alternatively, the **-t** option (described below) can be used to specify which file will be used.

Remember that each file system on a computer is an individual unit. Therefore, **fsck** must be run for each file system.

3. File System Consistency

In order to maintain the consistency of a file system, UNIX is very careful about how and when the disk is updated. When this process is disturbed, corruption of the file system can occur. How does this happen?

The most common reason why a file system becomes corrupted is that it is improperly shut down. This can often be a result of a power failure. It can also be due to operator error, such as pressing the reset button while the system is running or forgetting to use the **sync** command prior to resetting the system or turning it off.

Any actual hardware failure, especially of the disk itself or its controller, can also result in corruption of a file system.

Another thing to remember about corrupt file systems is that they become worse with time. A corrupted file system, if not repaired, will undoubtedly become more corrupted. The results can be disastrous.

4. What Can Become Corrupted?

One of the most common places for corruption to occur is the super-block. Every file system has a super-block which contains the most basic parameters about the file system which UNIX needs to have on hand at all times. For example, the size of the file system, the number of free blocks, and the number of free inodes. The super-block is stored in block 0 of the file system.

As UNIX needs the information in the super-block at all times, a copy of it is kept in the main system RAM memory. In order to keep the version of the super-block on the disk up to date, the super-block is periodically copied from the RAM to the disk. This operation is called *sync.*

On a system running UNIX in multi-user mode, you will notice that a process called *update* is always running. The job of update is to do a sync operation every thirty seconds to update the super-block. On some systems, the update interval may be more or less than thirty seconds, but the principle is the same.

The super-block is changed so often that if a system is powered down without performing a final sync, there is a good chance that the version of the super-block in RAM memory, the "current" version, will not be the same as the version on the disk which is now out of date. This is the most common form of corruption.

Any file, directory or device on a UNIX system has an *inode.* The inode is a block which describes the file and contains information such as the size of the file and the time of last modification.

The domains taking part in this form of addressing are known as *hierarchical* domains. That is, each domain is fully contained within another, and can be considered as a geographical mapping. At any one level, there is a domain to describe every area in the world. There are also non-hierarchical domains. For instance, all those sites taking part in the Australian Computer Science Network belong to the domain 'acsnet', and this domain crosses geographical boundaries. One use of non-hierarchical domains is to consider them as special interest groups, and one may, for instance, address messages to everyone within them. Thus the address

*.netnews

will deliver a message to every site claiming membership of the domain 'netnews'.

Inter-Network Gateways using Domains

Most of the problems with current addressing mechanisms for computer messages stem from the interfaces required at gateways between different networks. The ACSnet domain mechanism allows gateway handlers to be attached to the name of a domain at a node. Then instead of being delivered locally, a message addressed to such a domain will be passed to its handler. The handler may massage the message and its address to conform to the new network, and pass it on.

Thus the domain 'UUCP' is considered to include all the nodes that communicate via the **uucp** network. Those nodes with interfaces to the **uucp** network from ACSnet will attach a gateway handler to the domain 'UUCP', and pass messages from ACSnet to **uucp** via the handler.

From ACSnet, the gateways to **uucp** are identified by their declared membership of the domain 'UUCP', and messages may be routed to the nearest node supporting the gateway.[2] A simple example of an address that would make use of a domain gateway is

john.decvax.UUCP

Such an address form is an expediency, pending the introduction of geographic domain addressing on the other network.

Domain Routing Mechanism

The underlying transport mechanism of ACSnet knows how to deliver messages to *handlers* at *nodes*. Nodes are linked to neighbouring nodes to form a network. *Nodes* correspond to computers, and *handlers* embody protocols that can address users.

ACSnet implements domains by attaching a list of accessible domains to each node. The route to a domain is therefore the same as the route to the nearest node (in the network routing sense) in that domain. Nodes are the lowest level in the domain hierarchy, and can be considered as domains with one member.

Primary and Secondary Domains

While a node may belong to many domains, it is considered to have one *primary* domain which specifies the set of nodes from which this node will be directly addressable. Any other domains are *secondary* and specify all other domains whose members will be directly addressable from this node.

Some of the domains will be fully contained within others, as a Computer Science Department is contained within a University, which is contained within a country. These domains, one of which must be the *primary*, are considered to belong to a local hierarchy. All other domains to which the node belongs are considered to be non-hierarchical. The hierarchy nominates which domains are sub-domains, each domain in the hierarchical list being completely contained within its successor.

[2]Of course, the gateway handlers must be fairly intelligent about manipulating the **uucp** addresses to correspond to the alternate points of entry to an explicitly routed network, but that's another story.

The hierarchy is used to detect sub-domains in domain lists presented from other nodes, and to reject different domains with the same name. An obvious example of this problem arises where there are two local domains called 'cs', but one is Computer Science at Melbourne University, and the other is Computer Science at Sydney University.

Source Address Construction

As messages proceed through the network, they may cross domain boundaries. At each boundary, it is necessary to augment the source address so that the message (or a reply) may be correctly returned to its source.

The routing mechanism at each node notices when a new destination does not belong to the same primary domain as the source (or most recent node in the route), and appends an appropriate part of the local domain hierarchy to the source address. Every time a message arrives at a node, the source address is scanned from the right hand end, and each domain to which the local node belongs is removed. In this manner, at any one node in the route, the source address always contains the minimum set of domains to identify the source uniquely.

An example is given by the progress of a message addressed to a node 'snb' in the domain 'btl' inside 'usa' from the node 'psych23' inside the domains 'psych', 'su' and 'oz':

Node	Domains		Addresses	
	Primary	Hierarchy	Source	Destination
psych23	psych	psych.su.oz	psych23	snb.btl.usa
psych44	su	psych.su.oz	psych23.psych	snb.btl.usa
basser	oz	cs.su.oz	psych23.psych.su.oz	snb.btl.usa
research	usa	btl.usa	psych23.psych.su.oz	snb
snb	btl	btl.usa	psych23.psych.su.oz	

The source address is built up at 'psych44' from where the message crosses out of the domain 'psych' into the domain 'su', and at 'basser' from where the message crosses out of the domains 'su' and 'oz' into the domains 'usa' and 'btl'. Notice that as each destination domain is reached, it is removed from the destination address.

Routing Efficiencies

In practice, the introduction of domains has reduced the number of nodes visible at the highest level by a factor of two. With $O(n^2)$ routing algorithms, this is a great saving. Better still, for nodes at lower levels, the amount of routing information has been cut by up to a factor of ten, making network membership a much more affordable option.

There is another benefit in the area of network topology maintenance. A broadcast address is defined as one that delivers its message to every node in the primary domain of the sender (unless overridden by explicitly naming a domain for the broadcast). This mechanism considerably reduces network routing traffic, which uses broadcast addressing to distribute topology changes, as messages that previously were sent to every node may now be confined within the relevant domain.

An individual inode, or the main list of inodes for the file system, can also become corrupted. For example, each inode must be a file, a directory, a character device or a blocked device. If an inode does not have one of those four types, it is corrupted. Each inode must also be either in-use or on the list of free inodes. If an inode cannot be accounted for, it is corrupted.

Each inode which is allocated (in-use) must be linked to one or more directory entries. The link count, the number of directory entries for an inode, is one of the things which is stored in the inode and must match the number of actual directory entries found in the file system for that inode.

There is a list of blocks associated with every inode. If a block appears in the block list for more than one inode, it is considered to be a *duplicate* block. This is really a poor name because the block, itself, is not duplicated. In reality, the block is claimed by more than one inode.

Another possible corruption of the file system is a block with a bad number – either lower than the first data block number or higher than the highest. The list of free blocks on the system can also be checked for such an inconsistency.

The size of a file can also be checked. Each file has a 32-bit integer which contains the number of bytes in the file. This can be checked to see if the number of bytes is consistent with the number of blocks allocated to that file. Also, UNIX directory files can be checked to insure that the length is a multiple of 16 bytes.

Directories can be further checked to insure that correct links exist from the directory to itself and from the directory to its parent. Directories can also be checked for *connectivity*. This means that all files in the file system are accessible from the file system through its network of directories. If a file or a directory cannot be accessed, it is considered to be *orphaned*.

5. The lost+found Directory

If **fsck** finds a file or directory which has been orphaned, it needs a location where it can link that file or directory back into the file system. This is the *lost+found* directory.

The *lost+found* directory must always be created in the root directory of the file system. In the case of the first file system on a computer, this would be the system root directory ('/'). However, in the case of other file systems, the *lost+found* directory must be located in the root directory of that file system.

For example, if the file system */dev/dk2* (on Version 7 systems this might be */dev/rp3*) is mounted as the directory */u*, then a *lost+found* directory must be created under */u* because it is the root directory of *dk2*.

The *lost+found* directory is made using the **mkdir** command. Then some files are created under that directory and then removed. This is because it is advisable that **fsck** never have to write on virgin areas of the disk (you may be working on a bad disk).

If some files are created within the *lost+found* directory and then removed, the space taken up in the directory by those directory entries (16 bytes per file) is not released. That space remains a part of the directory, itself. Therefore, if an orphaned inode is linked into that directory, the 16 byte entry is written on known good disk.

The following procedure is recommended for creating the *lost+found* directory using */bin/sh*. This procedure should always be performed on a newly created file system.

```
login as root
# cd /                    (or cd /u, etc.)    //change to the root directory
# mkdir lost+found                            //make the lost+found directory
# cd lost+found                               //change to the lost+found directory
# for i in a b c d e f g h i j k l m n o p
> do
> > $i                                        //use a shell loop to create a
> done                                        //few null files
# rm *                                        //remove the created files
# cd /
```

If **fsck** finds an orphaned file and links it back into the file system, it has to give the file a unique name. Therefore, it simply uses the inode number, which is always unique, as the file name. If inode number 986 is recovered on file system *dk0*, it will be accessible when **fsck** is finished as */lost+found/000986*.

If a whole directory has been orphaned, it is relinked into *lost+found* as a directory and the file names under that directory will be correct.

6. fsck Run-Time Options

fsck has a number of run-time options. However, some are extremely dangerous and should be used with care.

fsck –[y|n][s|S] [–t *scratchfile*] [*filesystem* ...]

-y option:

The -y option tells **fsck** to assume that the answer to all operator questions will be "yes." This is an extremely dangerous option as **fsck** will move unconditionally through the file system. It should **NEVER** be used.

-n option:

The -n option tells **fsck** to assume that the answer to all operator questions will be "no." This can be used safely as a no answer to all questions means that the disk will not be written on at all, only read. It can be useful to use this option in order to see the extent of the damage to the file system before making changes.

-s option:

The -s option will force a new super-block to be created from scratch for the file system. This option should not be used except by qualified personnel.

-S option:

The -S option is similar to the -s option except that the super-block will be re-created only if **no** discrepancies are found in the file system. This is used for forcing the free list to be reorganized if the file system is operating fine. It also assumes the -n option so that no operator responses are necessary. If any discrepancies are found in the file system, the free list is not reorganized. There is usually no need for this option to be used.

-t option:

fsck uses main memory to keep the large tables that it needs in order to check the file system. If there is not enough memory space available, **fsck** will prompt the operator for a place to create a scratch file. If the -t option is used, the file named as the next argument will be used as the scratch file. The file used should not be on the file system being checked. If the file is a plain disk file which did not exist before, it will be removed when **fsck** is finished.

filesystem:

Following the above options, the user can specify the names of the file systems to be checked. If no list of file systems is given, **fsck** will use the list found in the file */etc/checklist* (if any).

7. fsck Error Messages – Initialization Stage

The possible error messages from **fsck** can be broken down into various stages. The errors in the initialization stage result from bad parameters or hardware problems.

C option?
: An option 'C' was given which is not one of the valid options (-y, -n, -s, -S, -t). See the section "Run-Time Options." If this error occurs, **fsck** will terminate immediately.

Bad -t option
: The -t option was used but was not followed by a file name. See the section "Run-Time Options." If this error occurs, **fsck** will terminate immediately.

Invalid -s argument, defaults assumed
: The -s option was not suffixed by a '3', '4' or values for the number of blocks per cylinder and the number of blocks to skip. The values which were used when the file system was created are assumed. If no such values were given, the values of 400 blocks per cylinder and 9 blocks to skip are used (400:9). See **fsck**(1) and **fsck**(8) in the UNIX Manual for more information.

Incompatible options: -n and -s
: It is not possible to change the free list without modifying the file system. Therefore, the -s and -n options can not be used at the same time. See the section "Run-Time Options." If this error occurs, **fsck** will terminate immediately.

Can't get memory
: **fsck** requested memory in order to create its tables. The request failed. This error should never occur. If it does, contact qualified support technicians. If this error occurs, **fsck** will terminate immediately.

Can't open checklist file: *filename*
: The checklist file could not be opened for reading. This is usually */etc/checklist*, and the file should be set to allow read access. If this error occurs, **fsck** will terminate immediately.

Can't stat root
: One of the first things which **fsck** tries to do is to get the information on the root directory. This message occurs if the attempt fails. This error should never occur unless you are trying to run **fsck** on a disk which has no file system. If it does, contact qualified support technicians. If this error occurs, **fsck** will terminate immediately.

filename is not a block or character device
: The name which you have given to **fsck** as the name of the file system to be checked is not correct. It is a plain file rather that a device. If this error occurs, **fsck** will ignore this device name and go on to the next one in the checklist.

Can't open *filename*
: The named file system could not be opened for reading. If this error occurs, **fsck** will ignore this device name and go on to the next one in the checklist. The permissions of the file system should be changed to allow read access.

Size check: fsize *file system size* isize *inode list size*
: **fsck** has found that the inode list has more blocks in it than the whole of the file system. Otherwise, there are more than 65,535 inodes in the file system. If this error occurs, **fsck** will ignore this device name and go on to the next one in the checklist.

Can't create *filename*
: **fsck** tried to create a scratch file for its tables with the mentioned file name. If this error occurs, **fsck** will ignore this device name and go on to the next one in the checklist. The permissions on the file should be checked.

Only the duplicate block error condition can result from phase 1B. You can manually determine from phase 1 and phase 1B which inodes contain overlapping blocks.

block number DUP I=*inode number*
This inode contains the given block number which is also associated with another inode. There is nothing which can be done in response to this message and if there are a lot of duplicate blocks, the EXCESSIVE DUP BLOCKS error will occur.

10. fsck Error Messages – Phase 2: Check Pathnames

In this phase, **fsck** removes directory entries related to bad inodes and then checks all directories for references to inodes which are out of range or not allocated. It also checks the root directory for correct mode and status bits.

ROOT INODE UNALLOCATED. TERMINATING
The root directory inode (usually inode number 2) has no allocation bits. As this should never happen, **fsck** will automatically terminate. If this error should occur, contact a qualified system technician.

ROOT INODE NOT DIRECTORY (FIX)
The root directory inode (usually inode number 2) was found not to be of the directory type. A **yes** answer will cause **fsck** to try to fix the type of the root inode. However, if the data blocks associated with the root inode are not found to have directory entries, a VERY large number of error messages will follow. A **no** answer will cause **fsck** to terminate.

DUPS/BAD IN ROOT INODE (CONTINUE)
Duplicate or bad blocks were found in the root inode (usually inode number 2) during phase 1 or phase 1B. A **yes** answer will cause **fsck** to ignore this error condition and to attempt to continue checking the file system. However, if the root inode is not correct, a large number of other error conditions may result. A **no** answer will cause **fsck** to terminate.

inode number OUT OF RANGE I=*inode number* NAME=*filename* (REMOVE)
A directory entry with the given name has this inode number which is out of range, either smaller than the first inode number in the file system or bigger than the last one. In other words, the directory entry is useless as it refers to a file which does not exist. A **yes** answer will cause the directory entry to be removed. A **no** answer will cause the error to be ignored.

UNALLOCATED I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*time last modified* NAME=*filename* (REMOVE)
A directory entry has been found with the given name, owner, mode, size, and last time of modification. The inode, however, has no allocation bits, meaning it is currently unallocated. This can happen when the system has only partially created a file when the system crashes. A **yes** answer will clear the directory entry which is in error. A **no** answer will cause the error to be ignored.

DUP/BAD I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*time last modified* DIR=*directory name* (REMOVE)
Duplicate or bad blocks were found in the directory entry for this directory inode during phase 1 or phase 1B. This means that the directory entry is probably corrupted. A **yes** answer will remove the damaged directory entry. A **no** answer will not.

DUP/BAD I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*time last modified* FILE=*filename* (REMOVE)
Duplicate or bad blocks were found in the directory entry for this file inode during phase 1 or phase 1B. This means that the directory entry is probably corrupted. A **yes** answer will remove the damaged directory entry. A **no** answer will not.

11. fsck Error Messages – Phase 3: Check Connectivity

In this phase, **fsck** checks to make sure that all directories are properly connected to the file system. It will find unreferenced directories and, if possible, connect them to the *lost+found* directory.

UNREF DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*time last modified* (RECONNECT)
A directory was found with this inode number which is not connected to the file system. A **yes** answer will result in the directory being connected to the *lost+found* directory. The inode number will be used as the file name. This may result in the DIR CONNECTED message below.

SORRY. NO lost+found DIRECTORY
There is no *lost+found* directory in the root directory of the file system or the permissions on the *lost+found* directory do not allow reading and writing. You have just made a request to connect a directory to *lost+found.* Your request will be ignored and the directory will remain unconnected. You will have to run **fsck** again with a proper *lost+found* directory in order to reconnect this directory. This error will also result in the UNREF error condition in phase 4.

SORRY. NO SPACE IN lost+found DIRECTORY
There is no space to add another entry to the *lost+found* directory. You have just made a request to connect a directory to *lost+found.* Your request will be ignored and the directory will remain unconnected. You should either remove unnecessary entries from the *lost+found* directory or make *lost+found* larger by creating empty files and removing them (see the section "The *lost+found* Directory" above). When this has been done, you will have to run **fsck** again in order to reconnect this directory. This error will also result in the UNREF error condition in phase 4.

DIR I=*inode number* CONNECTED. PARENT WAS I=*inode number*
This message advises you that a directory inode was successfully connected to the *lost+found* directory. Under the reconnected directory, the parent directory ('..') has been linked to the *lost+found* directory. The second inode number given is the inode number of the former parent directory of this directory.

12. fsck Error Messages – Phase 4: Check Reference Counts

During phase 4, **fsck** checks the link count table which was created during phases 2 and 3 and identifies incorrect link counts for files, directories and special files. It also locates unreferenced files.

UNREF FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* (RECONNECT)
The file with this inode number exists but is not connected to a directory (it is orphaned). If you respond **yes**, the inode will be connected to the file system in the *lost+found* directory. If you respond **no**, the error will be ignored; this will result in the CLEAR error condition below.

SORRY. NO lost+found DIRECTORY
There is no *lost+found* directory in the root directory of the file system or the permissions on the *lost+found* directory do not allow reading and writing. You have just made a request to connect a file to *lost+found*. Your request will be ignored and the file will remain unconnected. You will have to run **fsck** again with a proper *lost+found* directory in order to reconnect this file. This error will also result in the CLEAR error condition below.

SORRY. NO SPACE IN lost+found DIRECTORY
There is no space to add another entry to the *lost+found* directory. You have just made a request to connect a file to *lost+found*. Your request will be ignored and the file will remain unconnected. You should either remove unnecessary entries from the *lost+found* directory or make *lost+found* larger by creating empty files and removing them (see the section "The *lost+found* Directory" above). When this has been done, you will have to run **fsck** again in order to reconnect this file. This error will also result in the CLEAR error condition below.

(CLEAR)
Your request to reconnect this inode (which just failed) could not be completed. If you respond **yes**, the inode will be deallocated and added to the free inode list. If you wish to make another attempt with **fsck** to save the inode, you should respond **no**, and the error condition will be ignored.

LINK COUNT FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* COUNT=*link count* SHOULD BE *link count* (ADJUST)
This file inode has an incorrect count of the number of links to it. A **yes** answer will modify the link count with the correct value. A **no** answer will cause the error to be ignored.

LINK COUNT DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* COUNT=*link count* SHOULD BE *link count* (ADJUST)
This directory inode has an incorrect count of the number of links to it. A **yes** answer will modify the link count with the correct value. A **no** answer will cause the error to be ignored.

LINK COUNT *filename* I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* COUNT=*link count* SHOULD BE *link count* (ADJUST)
This file inode has an incorrect count of the number of links to it. A **yes** answer will modify the link count with the correct value. A **no** answer will cause the error to be ignored.

UNREF FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* (CLEAR)
A file was found with this inode number which is not connected with the file system and could not be reconnected. After a crash, this message often refers to an inode for a pipe (MODE=10000). In this case, the inode can be safely deallocated without concern. A **yes** answer will result in the inode being deallocated and the blocks allocated to that inode being released. A **no** response will cause the error to be ignored.

UNREF DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* (CLEAR)
A directory was found with this inode number which is not connected with the file system and could not be reconnected. A **yes** answer will result in the inode being deallocated and the blocks allocated to that inode being released. A **no** response will cause the error to be ignored.

BAD/DUP FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* (CLEAR)
Duplicate or bad blocks were found in this inode during phase 1 or 1B. If a file contains bad blocks, you may wish to try to read and salvage the file before removing it. If a file contains blocks also claimed by another inode, you may want to try and do the same. If two inodes claim the same block, usually only one of them is corrupted. This is usually the inode with the oldest date of modification. You should be selective in removing inodes under this error condition. A **yes** answer will deallocate the inode. A **no** answer will not.

BAD/DUP DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes*
MTIME=*last modification time* (CLEAR)
Duplicate or bad blocks were found in this inode during phase 1 or 1B. If a directory contains bad blocks, you may wish to try to read and salvage the directory before removing it. If a directory contains blocks also claimed by another inode, you may want to try and do the same. If two inodes claim the same block, usually only one of them is corrupted. This is usually the inode with the oldest date of modification. You should be selective in removing inodes under this error condition. A **yes** answer will deallocate the inode. A **no** answer will not.

FREE INODE COUNT WRONG IN SUPERBLK (FIX)
The actual count of free inodes in the file system does not match the count in the super-block. This is often because you have just removed some inodes and the free count has been changed. If you reply **yes**, the count in the super-block will be replaced with the correct value. A **no** answer will cause the error to be ignored.

ROOT INODE NOT DIRECTORY (FIX) ..... Phase 2
DUPS/BAD IN ROOT INODE (CONTINUE) ..... Phase 2
inode number OUT OF RANGE I=*inode number* NAME=*filename* (REMOVE) ..... Phase 2
UNALLOCATED I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 2
DUP/BAD I=*inode number* OWNER=*user id* MODE=*mode bits* ... DIR=*directory name* ..... Phase 2
DUP/BAD I=*inode number* OWNER=*user id* MODE=*mode bits* ... FILE=*filename* ..... Phase 2

UNREF DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 3
SORRY. NO lost+found DIRECTORY ..... Phase 3
SORRY. NO SPACE IN lost+found DIRECTORY ..... Phase 3
DIR I=*inode number* CONNECTED. PARENT WAS I=*inode number* ..... Phase 3

UNREF FILE I=*inode number* OWNER=*user id* MODE=*mode bits* ... (RECONNECT) ..... Phase 4
SORRY. NO lost+found DIRECTORY ..... Phase 4
SORRY. NO SPACE IN lost+found DIRECTORY ..... Phase 4
(CLEAR) ..... Phase 4
LINK COUNT FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
LINK COUNT DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
LINK COUNT *filename* I=*inode number* OWNER=*user id* MODE=*mode bits* ..... Phase 4
UNREF DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
UNREF FILE I=*inode number* OWNER=*user id* MODE=*mode bits* ... (CLEAR) ..... Phase 4
BAD/DUP FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
BAD/DUP DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
FREE INODE COUNT WRONG IN SUPERBLK (FIX) ..... Phase 4

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE) ..... Phase 5
EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE) ..... Phase 5
BAD FREE BLK COUNT ..... Phase 5
count BAD BLKS IN FREE LIST ..... Phase 5
count DUP BLKS IN FREE LIST ..... Phase 5
count BLK(S) MISSING ..... Phase 5
FREE BLK COUNT WRONG IN SUPERBLOCK (FIX) ..... Phase 5
BAD FREE LIST (SALVAGE) ..... Phase 5

Default free-block list spacing assumed ..... Phase 6

count files *count* blocks *count* free ..... Cleanup
****** BOOT UNIX (NO SYNC!) ****** ..... Cleanup
****** FILE SYSTEM WAS MODIFIED ****** ..... Cleanup

17. fsck Error Messages – Alphabetical Order

Here is a list of the **fsck** error messages sorted by alphabetical order. Where necessary, the messages have been truncated to fit on one line.

Bad -t option ..... Init.
count BAD BLKS IN FREE LIST ..... Phase 5
BAD/DUP DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
BAD/DUP FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
BAD FREE BLK COUNT ..... Phase 5
BAD FREE LIST (SALVAGE) ..... Phase 5
block number BAD I=*inode number* ..... Phase 1
count BLK(S) MISSING ..... Phase 5
****** BOOT UNIX (NO SYNC!) ****** ..... Cleanup

C option? ..... Init.
Can't create *filename* ..... Init.
Can't get memory ..... Init.
Can't open checklist file: *filename* ..... Init.
Can't open *filename* ..... Init.
Can't stat root ..... Init.
CAN NOT READ: BLK *block number* (CONTINUE) ..... Init.

CAN NOT SEEK: BLK *block number* (CONTINUE) ..... Init.
CAN NOT WRITE: BLK *block number* (CONTINUE) ..... Init.
(CLEAR) Phase4

Default free-block list spacing assumed ..... Phase 6
DIR I=*inode number* CONNECTED. PARENT WAS I=*inode number* ..... Phase 3
DIRECTORY MISALIGNED I=*inode number* (CLEAR) ..... Phase 1
DUP/BAD I=*inode number* OWNER=*user id* MODE=*mode bits* ... FILE=*filename* ..... Phase 2
DUP/BAD I=*inode number* OWNER=*user id* MODE=*mode bits* ... DIR=*directory name* ..... Phase 2
DUPS/BAD IN ROOT INODE (CONTINUE) ..... Phase 2
count DUP BLKS IN FREE LIST ..... Phase 5
block number DUP I=*inode number* ..... Phase 1
block number DUP I=*inode number* ..... Phase 1B
DUP TABLE OVERFLOW (CONTINUE) ..... Phase 1

EXCESSIVE BLD BLKS I=*inode number* (CONTINUE) ..... Phase 1
EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE) ..... Phase 5
EXCESSIVE DUP BLKS I=*inode number* (CONTINUE) ..... Phase 1
EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE) ..... Phase 5

****** FILE SYSTEM WAS MODIFIED ****** ..... Cleanup
count files *count* blocks *count* free ..... Cleanup
FREE BLK COUNT WRONG IN SUPERBLOCK (FIX) ..... Phase 5
FREE INODE COUNT WRONG IN SUPERBLK (FIX) ..... Phase 4

Incompatible options: -n and -s ..... Init.
Invalid -s argument, defaults assumed ..... Init.
filename is not a block or character device ..... Init.

LINK COUNT *filename* I=*inode number* OWNER=*user id* MODE=*mode bits* ..... Phase 4
LINK COUNT DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
LINK COUNT FILE I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
LINK COUNT TABLE OVERFLOW (CONTINUE) ..... Phase 1

inode number OUT OF RANGE I=*inode number* NAME=*filename* (REMOVE) ..... Phase 2

PARTIALLY ALLOCATED INODE I=*inode number* (CLEAR) ..... Phase 1
POSSIBLE FILE SIZE ERROR I=*inode number* ..... Phase 1

ROOT INODE NOT DIRECTORY (FIX) ..... Phase 2
ROOT INODE UNALLOCATED. TERMINATING ..... Phase 2

Size check: fsize *file system size* isize *inode list size* ..... Init.
SORRY. NO lost+found DIRECTORY ..... Phase 3
SORRY. NO lost+found DIRECTORY ..... Phase 4
SORRY. NO SPACE IN lost+found DIRECTORY ..... Phase 3
SORRY. NO SPACE IN lost+found DIRECTORY ..... Phase 4

UNALLOCATED I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 2
UNKNOWN FILE TYPE I=*inode number* (CLEAR) ..... Phase 1
UNREF DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 3
UNREF DIR I=*inode number* OWNER=*user id* MODE=*mode bits* SIZE=*size in bytes* ..... Phase 4
UNREF FILE I=*inode number* OWNER=*user id* MODE=*mode bits* ... (CLEAR) ..... Phase 4
UNREF FILE I=*inode number* OWNER=*user id* MODE=*mode bits* ... (RECONNECT) ..... Phase 4

First 1985 Software Distribution Tape

The latest software distribution tape has been distributed to all 1985 Institutional and Supporting members who have the necessary licenses and release forms on file, and whose 1985 dues have been paid. If you have not received your tape, contact the USENIX office.

Frequently, distribution tapes contain material covered by various AT&T and/or University of California licenses. In the case of these restricted distributions, a member must have a copy of his relevant AT&T and UC licenses on file and verified at the Association office before the tape can be sent. Whenever possible, and when the nature of the submission warrants it, we also try to make versions of licensed software available on the unlicensed distribution tape, with the restricted files excised. In this case, the user must, obviously, replace the missing code by writing his own.

Each Institutional and Supporting member is entitled to receive one tape per distribution, even though there are occasionally multiple tapes to which he or she might be entitled. In this case, additional tapes may be purchased for $75 each, providing the member has all necessary licenses on file and verified.

Contents of the 1985.1 USENIX Tape

The 1985.1 USENIX Distribution tape is available in one of two forms: (1) one having no licensing restrictions, or (2) 2.9BSD license required. The only difference between the two tapes is that the unrestricted tape does not include the submission from the University of Guelph, which contains material covered by a 2.9BSD license from the University of California.

The tape with the 2.9BSD-licensed material will be sent only to those members who have *filed a copy* of their 2.9BSD license with the Association office. (Note that even though you may have other AT&T UNIX licenses on file, we must still receive a copy of your 2.9BSD license before we can send you a tape.)

Submitter:	Rick Macklem
Affiliation:	University of Guelph, Guelph, Ontario
Contents:	Modifications to the Berkeley 2.9BSD distribution to provide support for the DEC PRO 350. Includes an "ra" driver for the RQDX1 controller.
License Restrictions:	2.9BSD

Submitter:	Joe Kalash
Affiliation:	U.C. Berkeley
Contents:	Ingres relational database software distribution, with many bug fixes.
License Restrictions:	none

Submitter:	Peter Langston
Affiliation:	Bell Communications Research
Contents:	The legendary "Langston Games Tape" at last sees the light of day. Mostly binary sources for both VAX and Sun. Contents include (but are not limited to): Beasts, Bog, Bolo, Convoy, Dune, Empire, Fast Food, Grid, Oracle, Race, StarDrek, Wander, and War.
License Restrictions:	none, but some games use 4.?BSD features.

Submitter:	Andrew Royappa
Affiliation:	Computer Science Department, Purdue University
Contents:	**fps**, a fast version of **ps** for 4.?BSD.
License Restrictions:	none, but useful only on 4.?BSD.

Submitter:	Brian Reid
Affiliation:	Stanford University
Contents:	**nu**, a program to help a UNIX system manager create, modify, delete, and destroy user accounts.
License Restrictions:	none

Peter Gross, USENIX Tape Editor

The 1985.1 USENIX Software Distribution Tapes were prepared by Peter Gross of the High Altitude Observatory at the National Center for Atmospheric Research.

Publications Available

The following publications are available from the Association office or the source indicated. Prices and overseas airmail postage charges are per copy. California residents please add applicable sales tax. Payments **must** be enclosed with the order and **must** be in US dollars payable on a US bank.

USENIX Conference Proceedings

Meeting	Location	Date	Price	Overseas Postage	Source
USENIX	Portland	Summer '85	$25	$25	USENIX
USENIX	Dallas	Winter '85	$20	$25	USENIX
USENIX	Salt Lake	Summer '84	$25	$25	USENIX
UniForum	Wash. DC	Winter '84	$30	$20	/usr/group
USENIX	Toronto	Summer '83	*sold out*		USENIX
UNICOM	San Diego	Winter '83	$15	$25	USENIX

USENIX Association
P.O. Box 7
El Cerrito, CA 94530

/usr/group
4655 Old Ironsides Dr., #200
Santa Clara, CA 95050

EUUG Publications

The following EUUG publications may be ordered from the USENIX Association office at the above address.

The EUUG Newsletter, which is published four times a year, is available for $4 per copy or $16 for a full-year subscription. The earliest issue available is Volume 3, Number 4 (Winter 1983).

The July 1983 edition of the EUUG Micros Catalog is available for $8 per copy.

13. fsck Error Messages – Phase 5: Check Free List

In this phase, **fsck** checks the free block list to insure that all of the block numbers are within range, that there are no duplicate blocks or missing blocks in the list, and that the count of free blocks is correct.

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE)
: There are more than 10 blocks in the free list with a block number which is less than the minimum block number for this file system or more than the maximum block number. A **yes** response will cause **fsck** to ignore the rest of the free block list and continue. This will also result in the BAD BLKS IN FREE LIST error below. A **no** response will cause **fsck** to terminate.

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE)
: There are more than 10 blocks in the free list which are claimed by an inode or are duplicated within the list. A **yes** response will cause **fsck** to ignore the rest of the free block list and continue. This will also result in the DUP BLKS IN FREE LIST error below. A **no** response will cause **fsck** to terminate.

BAD FREE BLK COUNT
: The count of free blocks in the free block list in the super-block is greater than 50 or less than 0. There is no operator response for this condition and the condition will result in the BAD FREE LIST error below.

count BAD BLKS IN FREE LIST
: This number of blocks were found in the free block list with block numbers which are less than the minimum block number for this file system or more than the maximum block number. There is no operator response for this condition and the condition will result in the BAD FREE LIST error below.

count DUP BLKS IN FREE LIST
: This number of blocks were found in the free block list which are claimed by an inode or are duplicated within the list. There is no operator response for this condition and the condition will result in the BAD FREE LIST error below.

count BLK(S) MISSING
: This number of blocks which are unused by the file system were not found in the free block list. There is no operator response for this condition and the condition will result in the BAD FREE LIST error below.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)
: The actual count of free blocks in this file system does not match the count of free blocks in the super-block of the file system. A **yes** answer will replace the old value with the correct number. A **no** response will cause the error to be ignored.

BAD FREE LIST (SALVAGE)
: During this phase, bad or duplicate blocks were found in the free block list or blocks were found to be missing from the file system. A **yes** answer will cause the free block list in the file system to be replaced with the correct list. The new free block list will also be ordered to reduce the time spent waiting for the disk to rotate into position. A **no** response will cause this error to be ignored.

14. fsck Error Messages – Phase 6: Salvage Free List

In this phase, the free block list is reconstructed. The only error message from this phase is caused by bad values for the "blocks to skip" and "blocks per cylinder" variables. See **fsck**(1) and **fsck**(8) in the UNIX Manual for more information.

Default free-block list spacing assumed

This message advises you that the "blocks to skip" value is greater than the "blocks per cylinder," either value is less than 1, or the number of "blocks per cylinder" is greater than 500. The default values of 9 "blocks to skip" and 400 "blocks per cylinder" will be used.

15. fsck Error Messages – Cleanup

Once the file system has been checked by **fsck,** a few cleanup functions are performed. The following messages may be displayed.

count files *count* blocks *count* free

This message indicates the file system which was just checked has the given number of files, blocks, and free blocks.

****** BOOT UNIX (NO SYNC!) ******

This message indicates that the root file system or a mounted file system was modified by **fsck.** If UNIX is not rebooted immediately, the work which **fsck** has done may be undone by UNIX (see Section 2 "Using fsck"). After this message has been displayed, you must reboot UNIX. This is usually accomplished by resetting the computer.

****** FILE SYSTEM WAS MODIFIED ******

This message indicates that the file system was modified by **fsck.** If this file system is currently mounted or if it is the root file system, UNIX must be rebooted immediately or the work done by **fsck** may be undone by UNIX (see Section 2 "Using fsck"). This is usually accomplished by resetting the computer.

16. fsck Error Messages – Listed by Phase

Here is a list of the **fsck** error messages listed by the phase during which they occur. Where necessary, the messages have been truncated to fit on one line.

C option?.....Init.
Bad -t option.....Init.
Invalid -s argument, defaults assumed.....Init.
Incompatible options: -n and -s.....Init.
Can't get memory.....Init.
Can't open checklist file: *filename*.....Init.
Can't stat root.....Init.
filename is not a block or character device.....Init.
Can't open *filename*.....Init.
Size check: fsize *file system size* isize *inode list size*.....Init.
Can't create *filename*.....Init.
CAN NOT SEEK: BLK *block number* (CONTINUE).....Init.
CAN NOT READ: BLK *block number* (CONTINUE).....Init.
CAN NOT WRITE: BLK *block number* (CONTINUE).....Init.

UNKNOWN FILE TYPE I=*inode number* (CLEAR).....Phase 1
LINK COUNT TABLE OVERFLOW (CONTINUE).....Phase 1
block number BAD I=*inode number*.....Phase 1
EXCESSIVE BLD BLKS I=*inode number* (CONTINUE).....Phase 1
block number DUP I=*inode number*.....Phase 1
EXCESSIVE DUP BLKS I=*inode number* (CONTINUE).....Phase 1
DUP TABLE OVERFLOW (CONTINUE).....Phase 1
POSSIBLE FILE SIZE ERROR I=*inode number*.....Phase 1
DIRECTORY MISALIGNED I=*inode number* (CLEAR).....Phase 1
PARTIALLY ALLOCATED INODE I=*inode number* (CLEAR).....Phase 1

block number DUP I=*inode number*.....Phase 1B

ROOT INODE UNALLOCATED. TERMINATING.....Phase 2

Report on the European UNIX System User Group Meeting

Palais des Congrès, Paris
1st – 4th April 1985

Peter Collinson
Secretary

Introduction

A report of a conference headed "April 1st" may lead you to be somewhat suspicious. But, yes, there really was an EUUG conference starting on that date. It was held in the Palais des Congrès in the centre of Paris.

There were three days of conference meetings, and Network Events Ltd. organised a three day exhibition which started on the second day of the sessions. Allowing a free day to look at the exhibition was a successful idea. It was a reaction to the problem at Cambridge where attendees at the conference had to choose to miss a session to go to the exhibition.

The programme chair for the meeting was David Tilbrook, in positively his last appearance in the rôle because he returns to North America later this year. I would like to take this opportunity to thank him for everything that he has done for EUUG in the last two years. I suspect that a high proportion of the speakers who have allowed their bodies to be subjected to the punishment known as "long distance air-travel" have come to Europe because of arm twisting (or the threat of constant abuse in the mail) from David. It also is true to say that most of the speakers have enjoyed the conferences and hopefully returned to their homes telling their collegues of the good time that they had. This makes it easier to get other people to contemplate a trip to Europe in the future. Anyway, David expended a considerable amount of nervous energy and electronic mail handling time in setting this meeting up. The result was, I think, a good set of talks.

There were also a number of changes in the overall format of the conference. First, speakers were given longer slots than at previous conferences. This meant that there were only two speakers per session, rather than the three which were squeezed in before. It had been felt that the short 20 minute talks in previous conferences did not give speakers enough time to get to grips with their subject. Actually, it also seems that 35 minutes is not enough time for some speakers. I suspect that people are just used to talking for an hour and have difficulty in pacing themselves for a shorter period. However, the longer talks were an improvement.

Second, this was the first officially multi-lingual conference; there were simultaneous translations into French, German and English. The original intention was to provide translation between French and English, but the German interpreter came as part of the package. I have never experienced simultaneous translation before, and I think it worked.

One French speaker was a bit confused when the audience burst into gales of laughter because the English interpreter had a coughing fit at about 90 decibels. The laughter continued when the interpreter asked, "Did I translate something funny?" One problem was that the conference was mostly in English. Linguistic chauvinism meant that many people did not use the headphones during completely English speaking sessions and were then totally unprepared for the inevitable question in French from the floor (I did this too). Still, the translation enabled non-English language speakers to talk at an EUUG conference, and this **must** be a good thing.

The third change to the normal course of events was to give about two minutes to the session chairs so that they could say something about their local UNIX user group.

This article is reprinted, with permission, from the EUUG Newsletter.

Many of the speakers supplied papers which were printed in a book entitled *Papers Presented at the EUUG Spring Meeting, 1985* which was handed out in the Registration Packs. Another innovation – proceedings were printed *before* the meeting! The book is available from the EUUG office for the very reasonable price of five pounds.

Oh well, enough of this drivel; what follows are the abstracts for the talks plus some comment from me. Thanks again to David for getting the abstracts and sending them to me. This was no small job! Getting abstracts from speakers is hard enough, getting the papers even worse, but David maintains that getting mail through my machine is the hardest thing which he was obliged to do for the conference.

Day 1 – Monday April 1st, 1985

As usual this wasn't Day 1 for me. I had already managed to clock up two nights in Paris because the cheapest air fare requires a stay over a Saturday night, and it seemed reasonable to pay a short visit to my brother who lives there. What he didn't tell me was that we were invited to a party. This would have been fine but we didn't get back to the flat until 3am. I will never be quite sure which 3am it was, since France is an hour later than the UK anyway, and the powers-that-be decided to change to daylight saving time at 1am on that Sunday morning.

Day 2 of my visit, the Sunday, was taken up with committee meetings of different varieties, a late meal, and fairly early to bed to try and get over the lack of sleep and the two hour time change. I was in bed by midnight local time but the body kept saying "this is much too early" and refused to allow me to sleep. Still... conferences do seem to be about lack of sleep.

After a rather disorganised pre-organised breakfast of session chairs and speakers, I managed to make it into the Salle Bleue in the Palais, grab my headset and find a chair at the front. After the last conference in Paris spent sitting on hard benches, these chairs were sumptuous and comfortable – just the thing for sleeping in ... good job I had to take notes.

9.43am Session Chair: **Michel Gien, CNET PAA/TIM**

Michel officially opened the conference and introduced the Chairman of AFUU, the French UNIX user group.

9.44am **AFUU**
Jean-Louis Berand, Chairman, AFUU

The French group has 250 members. The group has grown from 8 members to its present number in three years. From the outside, it does seem that the French scene is being driven along much more by industry and software houses than the gleaming towers of academia. The group is involved with a commercial publishing company (CXP) to produce a UNIX software catalogue.

9.49am **Conference Organisation and Logistics**
David Tilbrook, Imperial Software Technology Ltd.

David introduced the program chairs and some of the speakers who were not speaking until the last day.

10.07am **VLSI Assist in Building a Multi-Processor UNIX System**
Ian Johnstone, Sequent Computer Systems

Abstract

Multiprocessors have been of interest to computer scientists and designers since the first computers. Three factors have limited the commercial success of multiprocessor systems: entry cost, range of performance, and ease of application. Recent advances have removed these limitations, making possible a new class of multiprocessor systems based on VLSI components.

The requirements for constructing an efficient MP system are detailed, including: low level mutual exclusion, interrupt distribution, inter-processor signalling, process dispatching, caching, and system configuration. A solution to these problems is described and evaluated.

Comment

I think this talk was interesting because it represents the start of the new era of applying specialised VLSI chips to solving computing problems.

The main goal of the work was to provide a low cost, high performance, multi-processing system. The base model of the system which was adopted was a fairly normal one, with several CPU's attached to a system bus supporting memory and I/O devices. In Sequent's system, a special control bus parallels the system bus and is used by a VLSI device called a *System link and interrupt control* (SLIC). There are thus two busses in the system, a high performance bus for data and a specialised bus for control functions. This keeps each bus simple with lower cost and higher reliability.

The SLIC chip is a 6000 gate custom CMOS array.

☛ Coffee ☚

11.15am Session Chair: **Peter Collinson, University of Kent**

Well, my "news from" was information about the UK part of USENET, which I have fallen into administrating more through misfortune than design. The UK network is now around 55 map entries and is growing in leaps and bounds – new sites are turning up every day and our mapping administration system deals with at least one map update every two days. The network is really a star based on **ukc**. This is not good, and we are busy re-organising the topology.

This information did not raise a single laugh; which was to be expected because networks are intrinsically boring. They serve to enable people to communicate and should work with no problems. Pity they don't.

11.23am **Concurrent Processing in Ada and UNIX**
Henk Hesselink, Delft University of Technology

Abstract

Over the past years interest in programming languages supporting concurrent processing has grown. Some of these languages are extensions of existing ones, e.g. concurrent Pascal. Others were designed to support concurrent processing from the start. An interesting (if only because of its backing) example of the second group is Ada, which is likely to become a major language in the future.

At Delft University of Technology a group is working on DAS, the Delft Ada Subset, which implements most of Ada except for its concurrent processing features. It therefore seemed like an interesting exercise to compare the support for concurrent processing in UNIX and Ada and to see to what extent this part of Ada could be mapped onto UNIX.

It may seem a little odd to compare a programming language with an operating system. However, UNIX in combination with a programming language (especially C) offers a similar kind of programming environment to Ada. Ada has simply integrated into itself what in UNIX are separate system calls, in the process ensuring that such facilities are standardised. The comparison is therefore between Ada on the one hand and UNIX plus a programming language on the other. Because the

programming language was C, the combination is referred to as **cunix**. The version of UNIX discussed in this paper is Version 7, so as not to cloud the issue with features present in some of the more current versions of UNIX but not in others. Mention is made of the effect of some of these features on the UNIX view of concurrent processing.

11.45am **MIRANDA: An Advanced Functional Programming System**
David Turner, University of Kent

Abstract

There has been an upsurge of interest in functional programming in the last few years, and a great deal of progress has been made in the design of functional languages. The spread of these new ideas has probably been delayed by a lack of widely available high quality implementations. The MIRANDA system has been developed to make available to the UNIX community a modern functional programming language, which it is hoped will become a *de-facto* standard. The language draws its features from the earlier languages SASL, KRC and ML, and is embedded in an interactive system providing the tools needed to build large applications. The talk will give an overview of the main ideas in the MIRANDA programming language, and will discuss the way in which the nature of UNIX has influenced the design of the MIRANDA system and facilitated its development.

Comment

Looking further forward than the next release of UNIX is something which we should perhaps do more often. I have been keen to get David along to an EUUG conference for some time. He enjoyed himself, "functional programming conferences are usually so dull affairs," he told me after the conference. David's talk was an introduction to the subject.

Functional languages are not a new idea but a new style of language. Conventional programming languages such as FORTRAN or C are: (1) based on assignment statements and state changes, and (2) are sequential – there is explicit flow of control. There is evidence that (1) and (2) have harmful consequences and have lead to the current software crisis.

Functional languages are descriptive rather than imperative. There is no assignment statement. There is no explicit flow of control. Such languages are FP, pure LISP and SASL.

The main features are: (1) conciseness – programs are shorter and easier to develop; (2) transparency – the programs have good mathematical properties, and (3) parallel executability – it ought to be possible to execute in concurrent processing environments. It is this last point which has the hardware and VLSI designers jumping up and down because hardware behaves in this way.

MIRANDA is a modern functional language implemented under UNIX. The system provides a good programming language and an interactive environment for programmers. The main properties are: (1) the language is purely functional; (2) it allows infinite data structures by delaying evaluation; (3) it has a set abstraction, and (4) it has polymorphic strong typing.

David gave a few short examples, like the *quicksort* algorithm in around 20 characters.

He talked about the MIRANDA programming environment. The program works in interactive "desk calculator" mode with a set of equations being input in any order and a result requested. The compiler works in conjunction with a screen editor which is currently **vi**, but this is user selectable. There is an on-line reference manual which is menu driven. The system allows full access to the UNIX shell (which is also nearly a functional language), and there is a useful library of built-in functions.

The current implementation is not fast, but faster implementations are coming. Because of speed limitations some possible uses for the system are: teaching, research, providing a vehicle for formal specifications, and a system for fast prototyping.

☛ Lunch ☚

Everyone who had looked at my previous reports had supposed that this one would be about wine. Well, the presence of good wine needs no special mention in France. Instead, I think that this conference was about lunches, which were excellent.

2.35pm Session Chair: **Jean Wood, Digital Equipment Corporation**

Jean's "news from" piece was about Decus Europe, whose next meeting is 16th-20th September at Cannes. Slides of Cannes, beaches, boats and some parts of some bodies followed.

2.38pm **The Influence of the UNIX Operating System on the Development of Two Video Games**
Peter Langston, Bell Communications Research

Abstract

The Lucasfilm Games Group was established to explore ways of applying the technological and methodological expertise developed at Lucasfilm (principally for film production) in a new entertainment medium – video games. One of the major tools on which that expertise is based is the UNIX operating system. It is an interesting coincidence that this operating system's early development was heavily influenced by games and the interests of game designers.

This paper describes the development of the two video games "Ballblazer" and "Rescue on Fractalus!" and the ways in which UNIX software aided and influenced their design.

The work described here was done at Lucasfilm Ltd., San Rafael, California.

Comment

This was a video tape of what had been produced with a very large piece of software running on a UNIX system. Very large pieces of software are often written in LISP, and this was no exception. The programs run on an Atari games machine and are very small pieces of software in comparison. The Atari machine is a 6502 processor chip plus a four channel sound chip. The graphics were really good for the games, and I particularily liked the way the sound (some tunes and other noises) had been constructed to enhance the games.

3.14pm **3-D Computer Graphics, the UNIX Way**
Tom Duff, AT&T Bell Laboratories

Abstract

The UNIX text processing tools are successful because they are simple, yet powerful, single-purpose operators which consume and produce documents in a simple, universal representation. Pipes and a programmable shell allow the simple operations of the individual commands to be combined in powerful ways. The data structures and compositing operators described in "Compositing Digital Images" (by Thomas Porter and Tom Duff, in Siggraph '84 Proceedings) can be the foundation a similarly powerful image synthesis and processing environment.

Recently, I have extended the Porter/Duff model to handle anti-aliased full 3-D images, and am developing a collection of 3-D rendering tools based on it. Each of the tools is a simple, single-function operator that consumes and produces pictures in this common format. For example, the programs that render fractal terrain and quadric surfaces use unrelated (even incompatible) algorithms. Nevertheless, since the programs produce output in a common format which the 3-D compositing program also reads and writes, we can render scenes which contain both sorts of object. I will describe the data representation and its 3-D compositing operation. A short 16mm film will illustrate how this scheme allows simple special-purpose rendering programs acting in concert to produce rich images.

Comment

A stunning film produced by some small pieces of code. Tom's main message was to underline the fundamental tools notion of UNIX. A quote which I liked: "The good thing about UNIX is not the code but the thought those people had: to use small pieces of code to generate big things in big ways." Interesting how different people's work in totally different areas have such large parallels, David Turner's message of "small is beautiful" is fundamentally the same as Tom's.

☛ Coffee ☚

4.17pm Session Chair: **Daniel Karrenberg, University of Dortmund**

Daniel spoke a few words in un-announced German to test the interpreters. He then talked in English about the German UNIX† user group. The German group has 151 members. They have one meeting a year. The first one was in Frankfurt in February this year. They also produce a newsletter; the intention is to generate more than four a year.

4.21pm **Image Synthesis with UNIX**
Etienne Beeker, INA (Institut National de l'Audiovisuel)

Abstract

Realistic representations of three-dimensional scenes is a very attractive goal, object of research for several laboratories for the last few years. Our team at INA (the Institut National de l'Audiovisuel) has been writing such programs for two years, with the aim of creating special effects for video productions.

Two kinds of programs are now operational:

- Synthesis programs. These programs take a database of geometrical objects as input. The database consists of polygons and/or bicubic patches, and parameters such as colours, light sources, position of the virtual point of view, transparencies, etc. After the elimination of hidden areas and calculation of each pixel intensity with an appropriate illumination function, these programs produce as output an image in full colour, with shading, reflections, etc. One of them, using a scan-line algorithm, is completely operational. Other programs based on ray tracing and particle systems are still under development.

  These programs run under UNIX on a Perkin Elmer 3210. They require a lot of CPU though, and we expect a more powerful machine, like a VAX-785 or a Gould 3297.

- Design programs. These are more conventional CAD-like graphic programs. They allow the design of skeletons of objects in a wire frame, either with polygons or with mathematical surfaces like B-splines or bicubic patches. Geometrical smoothing of shapes is also implemented. An animation program interpolates object movement and the point of view. Movement control can be done quickly because of the wire frame display.

  These programs are running on a French 68000-based super-micro called SM90, with UNIX. Graphic output is done on a bitmap display. Coordinates are digitized with a tablet. These programs don't need much computation time but are highly interactive. That's why we run them on a machine used as a personal workstation. The database is then transferred to a more powerful machine for further calculations.

All our programs are written in C. The Perkin Elmer is running Edition VII (a V7 with some Berkeley enhancements), the SM90 is running UNIX V7.

Video production slides and animations produced with these programs will be shown, as well as a demonstration of the design programs running on a SM90.

†UNIX ist Warenzeichen der AT&T International.

Comment

The abstract sums up the talk well. It doesn't say that the system is being used for real to produce short "jingles" for French TV.

4.52pm **UNIX at IRCAM**
David Wessel, Robert Gross, IRCAM

Abstract

Since 1976 IRCAM (Institut de Recherche et Coordination Acoustique/Musique) has supported research on fast real-time digital signal processing, room and instrument acoustics, psycho-acoustics, compositional algorithms, as well as different methods of sound synthesis including the singing voice and physical modeling. The results of this research have been applied by invited composers in many contemporary music pieces, using real-time digital electronics and computer generated tapes.

IRCAM has developed a series of high speed digital signal processors culminating with the 4X machine recently used by Pierre Boulez in his work *Repons*. General purpose computing and program development is done on a group of VAX, SUN, Plessey, and Valid computers networked together. Work with artificial intelligence and expert systems has affected a majority of the current research projects.

Two years ago UNIX became the underlying foundation for all the research and musical production at IRCAM. UNIX must not only drive the 4X and support standard program development including numerical computation and compiler design, but also must do musical sample computation, storage, and real-time playback or record operations. This latter problem is not easy to deal with in UNIX and has been solved by using the advantages of UNIX files to simulate a hierarchical sound-file system based on the UNIX model. We are currently using the CARL (Computer Audio Research Laboratory) sound-file system developed at the University of California at San Diego.

This talk will explore some of the current development projects at IRCAM including the 4X software design, acoustical research, Chant/Formes project, applications on the array processor, and the development of music-oriented software for the Apple Macintosh using the SUMACC's (Stanford University) environment. We will also discuss our experience with the CARL software, and our attempts to utilize the 4.2BSD file system to support the I/O throughput required by the DA/AD conversion.

Comment

IRCAM is funded by the French Government through the Minister of Culture and the Minister of Research. They also have sponsors in Switzerland and the US. They have about 50 members on staff. Their work is entirely musical, and their aim is to aid composers to learn about the new technology.

Their computing environment is based on a 4.2BSD system running on a VAX-780. The machine has an array processor and D/A and A/D conversion systems. The machine is connected via Ethernet to some workstations which are used for graphical input. They deal in large volumes of data. For instance, two seconds of 16 channel 32 Kb/sec digital data for the converters occupies two cylinders on an RM05.

There followed tape recordings of some of the stuff which is being done. One was a totally synthesised singing human voice; it was somewhat eerie. Another was the output from a program which modelled the physics of a violin. The original program sounded like a beginner because they hadn't programmed the bowing action properly. A later example of synthesised jazz violin was terrific.

The session finished with a video tape of a system which connects a flute player with an automatic accompanist. First, the piece was played straight and the accompanist program followed it. Then, the player tried to put the program off: he made changes of timing and played "badly" by missing notes or playing wrong ones. The accompanist followed. I suddenly had this vision of a small box which buskers on the Tube take with them to accompany their efforts.

All impressive stuff, and this was followed on the last night of the sessions by a concert at IRCAM itself.

☛ End of Day 1 (??) ☚

No, it wasn't. The formal meeting was immediately followed by a small session called "Birds of a Feather sessions." Actually, they were short presentations.

5.35pm **The SM90**
??

Sorry, I didn't get this person's name and I grovel most humbly. I think that he was from IRCAM.

This was a brief description of a UNIX system running on a multiprocessor machine – the SM90. The usual pattern of a shared system bus is repeated here, but every CPU has its own local bus supporting memory and I/O devices. The global memory is used for system tables and local memory in each processor contains user processes and system code. The selection of which processor is to be used by a process is made by the user.

5.58pm **The Network File System**
Brian Novak, Pyramid

Pyramid has adopted Sun's network file system (NFS) for their machines. This talk was a brief introduction to its use.

The design goals of the NFS were: transparent remote file access, a simple implementation, the use of a general purpose protocol, and the implementation of a fast file system.

The system does not attempt to be a distributed UNIX system, nor was it designed to provide access to devices on the network or file locking.

The NFS uses a stateless protocol which makes easy recovery from temporary interruptions. Local file caching gives the system good performance. It is possible to connect file systems running on architecturally different machines because the protocol has a canonical form of data representation.

To my mind, the hardest thing to get right on these systems is the area of file security and access. The NFS insists that a user has a unique **uid** across the network, and I think this is entirely unrealistic. The system prevents super-user access across the network, but this is just one obvious thing to do. In my experience, potential system busters look first at the people who are able to become super-user. Still... no doubt the system works.

6.15pm **UUCP Matters**
Various

How this got onto the agenda, I am not sure. When the moment came, there was no one to speak to it, so I got lumbered. I did not think that much came of the meeting, but we did manage to air the problem of funding mail on the European network. There are no real solutions; unfortunately we are all running software which was intended to run without accounting or controls.

☛ End of Day 1 (it really was, this time) ☚

And onto a wine reception paid for by UNIX Europe, for which many thanks. Unfortunately, I had to leave and go to another UUCP meeting, this time with backbone administrators. That is best glossed over – except that it seemed to be held in a sauna because we were unable to succeed in opening the window in the room and the temperature soared.

Day 2 – Tuesday, 2nd April

9.36am Session Chair: **Peter Langston, Bell Communications Research**

Bell Communications Research is an R&D department which is funded by the local telephone companies in the US. It should not be confused with Bell Laboratories, which is the research arm of AT&T. Peter talked about the trivia at the last USENIX meeting. It was interesting that the program that placed second in the GO tournament was one which contained no strategy, but just did random moves. The next USENIX meeting is in Portland, Oregon, starting on June 12th.

9.43am **VIDMAN: For a New Step Towards a Better Communication**
Didier Galmiche, Laboratoire d'Informatique Theòrique et Programmation

Abstract

VIDMAN is new software running on a 4.2BSD UNIX system that enables the creation of a visual manual. First of all, we should say that this product was necessary to enable the "broadcasting" of our research works. The best encouragement we have received is the unanimous support from LITP's members.

Our main idea was to build up a manual that could improve at the same time as its contents. As a matter of fact, continuous updating is quite impossible with a manual written on sheets of paper, whereas it becomes almost natural with VIDMAN.

We think that this is a useful tool which is easy to use and that it offers a new way of communication for UNIX users. Moreover, VIDMAN can be used to describe any sort of information: lectures, new languages and theories, description of systems, etc.

Thanks to judiciously chosen controls, it offers unusual freedom in document styling and consultation. **nroff**-style commands are used for information formatting, creating paginated text having a user-defined style. For inspection of the information, **emacs**-style key definitions allow access to the desired information. Three types of pages can appear on the screen: menu pages (index pages with lists of the stored items), text pages (information pages corresponding to each item), and mixed pages containing the two last types in any proportion.

VIDMAN can prepare output for all terminal types and allows the utilization of the full resolution of each terminal, thanks to the *termcap* database.

Finally, we must say that VIDMAN is written in the C programming language, and the use of some libraries of programs such as **curses** and **termlib** and the use of **lex** (Lexical analyser generator) is essential. This means that VIDMAN is specific to UNIX.

Comment

Another system done for a French TV channel. The system is aimed at efficient manipulation of an on-line manual. The data is tree structured, where the leaves are display data and the nodes either are shell commands or point to leaves.

The system is in use at LITP and IRCAM.

Q. Can you store an update history?

A. Yes, this should be possible but isn't currently implemented ... that's a good idea ... we had better do that.

Q. Is it available?

A. Yes, in a limited form.

10.10am **System Aspects of Low-Cost Bitmapped Displays**
Dave Rosenthal, Information Technology Center, Carnegie-Mellon University

Abstract

The design of low-cost bitmapped displays is reviewed from the perspective of the implementors of a window manager for UNIX. The interactions between RasterOp hardware, the multi-process structure of UNIX, and the functions of the window manager are discussed in the form of a check-list of features for hardware designers.

Comment

Dave started off by talking about what is happening at CMU. The Information Technology Center (ITC) at Carnegie-Mellon has the aim of generating a workstation-based environment. Every student who comes into CMU will be required to buy a workstation which will have 1 megabyte of memory, perform at 1 Mip, and have a bit-mapped display and a mouse. All the machines on the campus will be linked by a high bandwidth local area network, and the workstations will have access to a network file system which will encompass all the machines on the campus. The workstations are to run 4.2BSD, and the machine currently in use is the Sun workstation. This may change because the system is being funded by IBM. ITC currently has about 30 people and has a lot of hardware.

The network is very large. Currently every research and teaching room has Ethernet or Pronet connections, and there are about 500 taps into the net. All the campus machines (6 DEC-20's, about 70 VAX's, some HP machines, and of course some IBM machines) use the DARPA IP protocols. There are plans to have at least 8K taps onto the network by 1987. It is proving expensive to wire the campus. The projected cost for wiring every room is in the region of $8 to $10 million.

The network file system provides each workstation with global file access to a large UNIX tree-structured namespace. Currently, there are two file systems in production, one with 50 clients on four servers and one with 80 clients on six servers. A total rebuild of the system is taking place; this is based on the results of the current versions.

The user interface to the system on the workstations uses bit-mapped graphics. ITC has developed a portable way of addressing the bitmap display, giving window access from any machine in the network, not just the workstations. The interface allows easy use of UNIX from menus, icons, etc. Users have uniform access to high quality multifont text. The interface is currently running on a network of about 100 Suns. The software drives both the Sun1 and the Sun2 hardware and avoids the built-in Sun RasterOp hardware. The software requires no kernel modifications. ITC seeks to encourage development of educational applications with a uniform, easy-to-learn interface.

The window manager is a user level server process, unlike Sun's. Clients do remote procedure calls over TCP/IP stream connections, so clients can access a window on your workstation from anywhere on the network. Output is via lines, multifont text, and RasterOps. The window manager process tracks the mouse, implements menus, and demultiplexes keyboard input.

After all that, Dave managed to get onto talk about his paper. I quote the concluding paragraph: "We have set out a number of points worthy of consideration in the design of low-cost bitmap displays intended to support multi-process interaction. Although RasterOp hardware may appear attractive, it needs careful design if its potential is to be fully realised, particularly for passing characters. Assistance with cursor drawing and sharing of the colour map may be more cost effective uses for limited hardware resources."

Q. Can we get the software?

A. Yes. The window manager, base editor and other software is available on a tape to selected (mostly academic) sites. The tape also contains some public domain screen fonts derived from the TEX fonts. The address to contact is:

Distribution Co-ordinator
Information Technology Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

☛ Coffee ☚

11.20am Session Chair: **Jim McKie, Centrum voor Wiskunde en Informatica**

Jim introduced Johann Helsingius, who talked about the Finnish UNIX User group or FUUG. Johann started in Finnish to confuse the interpreters and later refused to translate what he had said – perhaps the conference competition should have asked for some guess about the content of his initial sentence.

FUUG has 80 members, but they expect the membership to grow because they have not followed up contacts made at an initial meeting. The proceedings of the meetings are in Finnish. There are about 20 UUCP sites in Finland. The main problems are concerned with character sets: the characters '{', '}', '[', ']' and '\' are used for special characters; text written in Finnish looks like C to the **file** program.

11.27am **UNIX Networking Via X.25**
Radek Linhart, Hewlett-Packard GmbH

Abstract

X.25 packet switched networks provide a far more reliable and cheaper data transfer method than asynchronous links via telephone lines. This talk will give a brief overview of the typical features provided by national X.25 networks. In addition, different alternatives of network access will be discussed.

Finally, a Hewlett-Packard company internal implementation will be introduced. A flexible, easily configurable approach with a high degree of security has been taken.

Comment

This talk was a good overview of the communication problems associated with running the UUCP network. Radek wished to connect the European Hewlett-Packard sites into USENET, which is used for a considerable amount of internal company mail and information.

He came to the conclusion that, for European sites, X.25 connections were 8 times cheaper, 8 times faster and 100,000 times more reliable than telephone connections. It is also considerably more expensive for US sites to get X.25 connections than it is for European sites.

The system they have designed uses standard UUCP protocols (and Piet Beertema's f-proto) and has a program which allows high level programming of X.25 PAD parameters.

11.55am **ACSnet – The Australian Alternative to UUCP**
Piers Lauder, University of Sydney, Dept. of Computer Science

Abstract

ACSnet is a network with goals to serve a function similar to that currently served by the UUCP network. Routing is implicit, and addressing absolute, with domains. The network daemons attempt to make use of full available bandwidth on whatever communication medium is used for the connection. Messages consist merely of binary information to be transmitted to a handler at the remote site. That handler then treats the message as mail, news, files, or anything else. Intermediate nodes need not consider the type of the message, nor its contents.

Comment

Piers started by talking about the Australian UNIX system Users' Group – AUUG. It has around 250 members and around 40 network sites. The next meetings are in Brisbane on the 26/27th August and in Perth in February, 1986. Piers then talked about his paper.

ACSnet is the Australian alternative to the UUCP network and has obviously benefited from being designed with the notion of wide area networking rather than this function being grafted on in an *ad-hoc* way at a later date. The basic requirement was to have the ability to send a message from one point to another. A message is not simply mail but can be any sequence of bytes used for any purpose. The network is constructed from a set of nodes connected by any medium capable of connecting machines – phone lines, Ethernet, X.25, twisted pairs, etc. The system runs better over 8-bit data paths, but this is not mandatory. Any link may fail during transmission, and care is taken to restart the transmission where it left off, rather than restarting from scratch.

Addressing is domain based, which is a better model than the routing address used by UUCP. Messages may be broadcast to all machines on the net or in a sub-domain. Users can also explicitly route messages using "bang notation"; this is not strictly necessary but has proved useful for testing and loopback messages.

OK folks, let's all throw UUCP away....

☛ Lunch ☚

2.30pm Session Chair: **Teus Hagen, ACE**

Teus talked about the Netherlands UNIX User group, NLUUG. They held a meeting in March which 150 attended. The content was a "technical sales talk." Old hands thought that it was dreadful, but the audience liked it and asked for another. The next meeting is in the last week in November, and will be a single day with parallel sessions consisting of technical talks, talks from distributors, and an exhibition. The meeting will be in the Congres Center in Utrecht. The Dutch group is also involved in producing the UNIX Products Catalogue, which will become an EUUG publication, and of course, is also a prime mover of EUNET.

2.39pm **Addressing in MMDFII**
Steve Kille, University College, London

Abstract

The Multi-channel Memorandum Distribution Facility (MMDF) is a powerful and flexible message handling system for the UNIX operating system. It is a message transport system designed to handle large numbers of messages in a robust and efficient manner. MMDF's modular design allows flexible choice of user interfaces, and direct connection to a number of different message transfer protocols.

This paper is intended as a sequel to the paper presented by Doug Kingston at the 1984 USENIX meeting in Salt Lake City. A brief technical overview of MMDF is given, and then two aspects are considered in more detail:

(1) Address handling. The table-driven approach taken by MMDF to handling a structured address space is described. The extension of this approach to use with distributed nameservers is considered.

(2) Message reformatting. A number of systems using similar, but distinct, message and address formats have emerged. The approach taken by MMDF to allow interworking is described.

A comparison with other systems, in particular **sendmail**, is made.

Comment

MMDF was initially developed in 1979 by Dave Crocker to supply a mail system for CSNET. A production system was running in 1980. The new version was started in 1982 to take advantage of some of the new ideas from **sendmail** and also to take account of various alterations in protocols, specifically ARPA 822 and the peculiar UK protocols.

The goals (or perhaps features) of MMDFII are: robust and efficient message relaying; the support of multiple message transfer protocols; the ability to deal with several user interfaces; authentication of messages; and authorisation control on basis of sender and/or recipient. A big feature is the submission time address checking, which allows clean user interface design and minimises address transformation.

Steve then delved deep into the addressing mechanism, which is best left to his paper.

3.07pm **What is an International UNIX System?**
Duncan Missimer, Hewlett-Packard Company

Abstract

UNIX system users overseas are getting tired of talking to UNIX systems in whatever language it is that UNIX systems speak. Some people have responded by hand-crafting variants that speak the local language and properly handle exotic character sets. Others have created sub-environments that support different languages but do not offer the full functionality of the UNIX system. The authors believe that it is possible to create a fully international, fully functional UNIX system which can assume more than one linguistic identity at run time.

This paper presents a model for language and country custom independent software, which we use to outline some of the ways in which a vanilla UNIX system is unsuitable for use outside the English-speaking research environment.

We define two terms – native language support (NLS), and localization – and show how these concepts can be used to design an international UNIX system by moving language or country custom dependent information out of the source code and into the file system where it belongs.

Comment

The talk really pointed to the problem areas for linguistic change for utilities which run on the UNIX system. The problem areas are:

1. The system assumes the use of 7-bit ASCII, and often the eighth bit is used for specialised functions in programs.
2. Collation sequences in UNIX are based on the ASCII numeric sequence. This is not even adequate for American dictionary order, and much less for a language like Welsh where two symbols are used to represent a single character ('ll', 'ff', 'rh' etc).
3. Directionality, the assumption that languages go from left to right, is plumbed into the system.
4. Classification of characters in programs assumes a seven bit character set; this is insufficient for other varieties of character sets.
5. In a multi-lingual environment, standard escape sequences will most likely be used to indicate a change of character set. Software which processes these sequences may have to dynamically alter its treatment of the characters in any of the ways discussed above when an escape sequence is found.
6. Hyphenation and spelling – currently, **spell** and **nroff/troff** hyphenation support English only.
7. The computation and display of the date and the time is very US specific.
8. The names of the days of the week and the months are in English.
9. The names of currency units and ways of subdividing currency units require alteration.

10. There is significant world-wide variation in the representation of numbers, variation in the symbol used for the radix character, and variation in the symbol used for grouping digits, as well as the number of digits grouped.
11. Error messages, prompts and responses to prompts, and mnemonic command names should all be based on the user's language.
12. Messages built up in chunks from programs may have to alter in order when translated into another language.

Well, that list was taken from the paper. I thought that it was useful to include it here so that the magnitude of the problem can be envisaged. The main solution seems to be to comb through the code removing language dependency. HP's idea is to use a *termcap*-like language specification file and access the file from a variable in the environment.

3.29pm **Internationalisation of UNIX**
Gary Lindgren, UNIX Europe

AT&T is also addressing the problems mentioned above and wishes to provide a framework and tools for supporting local character sets, error messages in the local language, and a multi-lingual help facility.

There are plans to remove the eighth bit usage in editors.

There are plans to enhance the operating system, utilities, languages, and applications software to support international requirements in the areas of date and time format, collating sequences and numeric representation.

The System V Interface description contains a preliminary specification for the support of 8 bit and 16 bit character sets. This uses the top bit to indicate which character set is in force, coupled with a special shift character to indicate changes from one character set to another. My suspicion is that the scheme was a little complicated and would not really work. This whole subject seems to be a can of worms and very difficult to tackle in any easy way – but at least some people are trying.

☛ Coffee ☚

4.17pm Session Chair: **Sunil Das, City University, London**

Sunil is the chairman of UKUUG, the UK chapter of the EUUG. The group has 337 members and is the largest in Europe – for historical reasons. The main objectives of the group are to provide a UNIX information service, to provide a forum for discussion in the UK, to hold one-day technical sessions, and to aid and support UKC's efforts in administering UKNET.

4.24pm **Greek Characters on UNIX**
Steve Hull, Research Centre of Crete

Abstract

A case study is presented of a project to provide full Greek alphabet capabilities on a UNIX system. The particular difficulties of the Greek language in its various forms are discussed, as well as the full ramifications of providing "full capabilities" for a variety of hardware. The necessary tradeoffs are discussed, their limitations and advantages evaluated, and alternate approaches compared. It is hoped that, in addition to illuminating the particularly European problem of providing multilingual capabilities, we illustrate a worthwhile approach to solving UNIX problems in general.

Comment

The title of this talk was given as Ελληνικοœ Χαρακτήρες στο ΙΟΤΝΗΞ.† This was a really interesting talk which illustrated the complexities of the written Greek language and proposed some solutions to the problem. Again, I cannot do the paper justice here.

4.57pm **The SINDEX Chinese language project**
Paul Thompson, SINDEX

Abstract

The problem of Chinese text input derives from the large number of Chinese characters (10 to 20 thousand) and the way they are mapped on to a small number of monosyllables (approximately 1200). The solution is: (1) to avoid direct input of characters (i.e., the requirement to specify uniquely the character intended as we are accustomed to doing in western languages); (2) to input syllabic units phonetically; and (3) to use the linguistic constraints on syllabic combinations to convert the phonetic input into Chinese characters.

This solution has been implemented on UNIX, where the script conversion program has been written in C and the UNIX tools have made it possible to manipulate the relatively large linguistic database efficiently.

This talk will briefly discuss the problems and other solutions and will concentrate on the the use of linguistic constraints in our solution.

Comment

The talk started with a short piece of videotape from a UK TV programme showing the system in operation. People involved in the mechanical reproduction of Chinese usually ask about the speed of operation of the equipment being used and not the speed of the typist. Speed is not a problem with Paul's system. He has utilised a phonetic representation of Chinese which is taught in schools before writing in ideograms is taught. The system recognises the phonetics and makes a best guess about what character is to be output. It turns out that a high success rate is achieved by looking at the combinations of syllables and making judgements about them from their order.

5.20pm **EUUG Annual Meeting**
Teus Hagen, EUUG Director

Abstract

This is the official general meeting to present the new constitution.

Comment

Teus had several matters to report. The first was to note that Emrys Jones has resigned as Director of EUUG after three years' toil. The Executive Committee has decided to award him an honourary membership. Teus was elected Director of the EUUG at a Governing Committee meeting.

EUUG now has member groups in the UK, Eire (just thinking of breaking away from the UK group), Norway, Sweden, Finland, Denmark, Netherlands, Germany, Belgium (just breaking away from the Dutch group), France, Switzerland (looking at EUUG to see whether it is a good thing to join), Austria, Italy and Greece. Some countries in the Arabic world are also thinking of joining.

The national groups all elect two representatives to the Governing Committee, who in turn have control over the actions of the Executive Committee (who pretend to do the real work). The current Executive Committee is: a Director, Teus Hagen; a vice-director, Michel Gien; a secretary, Peter Collinson; a newsletter editor, Jim McKie; a financial manager, Mike Banahan; and a member without any special responsibility, Keld Simonsen. The business manager is Helen Gibbons.

†ΙΟΤΝΗΞ is not a trademark of AT&T Bell Labs.

The constitution had been voted on by postal ballot. There were 150 replies: 147 of these were "for," 2 "against," and 1 abstained. The constitution has thus been accepted by a fairly resounding majority.

Teus spoke of the various issues which were causing discussions in the various committee meetings, and also amongst the several committee members in bars (or perhaps one should say café's) in Paris.

I suppose that the main topic of concern is the issue of what sort of conference should be held. Is it important to have large externally organised exhibitions? When the exhibitions were organised by ourselves they used to subsidise conferences to a great extent, and now they raise very little revenue. They seem to cause the conferences to be held in expensive locations which in turn cause conference attendences to drop. Should the conferences be shorter? Longer? Should the conferences contain more sales promotion talks? Should the conferences include tutorials? Should we set up standard SIG's to provide smaller groups for meetings at conferences? We should raise some of these issues for discussion on the network. This, of course, does not get to everyone – perhaps people should submit their views for publication in the newsletter.

The newsletter is another issue for discussion. The fundamental problem is lack of copy and general input from the membership. There seems to be some contradiction because in general, members say that this is a useful thing for EUUG to do. However, there have been newsletters which have consisted totally of contributions from me. This is not healthy and eventually I will run out of asinine jokes.

The next conference will be in Copenhagen from September 10th to the 13th. There will be technical sessions from the 11th to the 13th, industrial sessions on the 10th and 11th, and an exhibition running from the 10th to the 12th. One of the guest speakers at the technical sessions will be Brian Kernighan.

☛ End of Day2 ☚

I think that there was a short "Birds of a Feather" session which I managed to miss (sorry). I realised that I had been inside for some time and had missed the fine Parisian weather. I started a movement to walk down to the Seine for the conference dinner, which was held in a boat called a "Bateau Mouche" travelling up and down the river. A number of passes were made past the Statue of Liberty, which had shrunk a bit. After a large quantity of alcohol, I found myself walking back to the hotel and managed to get involved in some late night discussions about this and that. And so to bed.

Day 3 – Wednesday, 3rd April

Up betimes and after a leisurely breakfast into a Governing Committee meeting. The first session of the morning had been cancelled because Mike Banahan had to go home for family reasons and Mike O'Dell was forced to cancel at the last moment because of some mess up with air tickets. There was some plan for Mike O'Dell to give his talk over the telephone, but I am convinced that this was formulated as an April Fool's joke by a certain big Canadian. I do hope that we will manage to get Mike to come to another conference. Anyway, the first session was cancelled. I think that something to do with UNIX Europe Ltd went on in the Salle Bleue; but as usual I found it difficult to be in more than one place at once. So there is no blow-by-blow commentary of the events.

11.15am Session Chair: **Bjorn Eriksen, ENEA DATA Sweden**

The morning started (or continued, depending on your point of view) with a few quick "news from" sessions.

11.17am **Indre By-Terminalen, Kobernaven Universitet**
Keld Simonsen

Keld was supposed to have chaired the cancelled session, so he got his chance to talk about the Danish Group. The Group was started in November 1983 and now has 91 members. The members are the usual mix of vendors, commercial users and academics. They put a small leaflet into most places where UNIX is sold in Denmark, which is a good idea, I think. The group produces two newsletters a year and has also published a UUCP installation guide. The network in Denmark has 12 to 15 mail-only sites and 3 to 5 sites taking news.

The group is naturally interested in international action on character sets and is involved in starting a group discussing "internationalisation"[†] of UNIX.

Denmark is the host of the next EUUG conference.

11.23am **The Norwegian Group**
Tor Jorgen Lande

The Norwegian group was started in May 1984. UNIX has not spread very far in Norway. They intend to hold one meeting per year and produce an annual newsletter.

11.24am **News from Sweden**
Bjorn Eriksen, ENEA Data

Bjorn finally got on to say his bit about the Swedish group – EUUG-S. He kindly gave me his overhead projector slide, so at least this bit of the report is accurate.

March 21, 1983: The group was started when Dennis Ritchie visited Sweden. There were about 40 members to start with and there are now 97. There are 28 individual members and 69 institutional members; of these 17 are educational sites and 80 are commercial.

Oct. 19, 1983: The first annual meeting, the invited speaker being Teus Hagen.

Sept. 26, 1984: Second annual meeting. David Tilbrook couldn't make it so they had Mike O'Dell and Mike Banahan.

March 1985: Presentation of Sun workstations.

May 1985: Meeting cancelled. David Tilbrook was going to explain the "Tilbrook Philosophy" but couldn't make it.

June 1985: Presentation of ULTRIX.

Sept. 1985: Next annual meeting. David Tilbrook is in Canada so they have Brian Kernighan instead.

Well, after all that, which didn't take long, onto the scheduled business of the morning.

[†]This was definitely the in-word of the conference.

11.27am **A Contractual Model of Software Development**
Vic Stenning, Imperial Software Technology

Abstract

The technical process of software development and indeed, development of any complex man-made system, can be viewed as repeated application of a single step. Each individual step takes some existing representation (or mathematical model) of the desired system and transforms it into some more "concrete" representation. This results in a sequence of representations leading from some very abstract model of the desired system to an actual implementation (i.e. a representation that is usable for the real world application).

Any practical approach to system development must address not only the technical issues, but also issues of project and data management. Thus the technical process outlined above must be incorporated into some broader project organisation. One possible approach is to employ a so-called "contractual" model of development, whereby contracts are let internally within the project for the performance of the individual transformation steps and any other work which needs to be done.

Individual contracts can be fulfilled by the use of appropriate technical development and project management methods. Data management methods must be employed both within individual contracts and at the level of the contract hierarchy for a complete project. An integrated project support environment can support the chosen methods within the framework of the contractual model.

Comment

Vic related a tale from a conference on Systems Design and Implementation where Freedman was discussing the analysis of problems within projects. He said, "Problems which were actually encountered during the design and implementation of systems ... only one problem was common to every system," and then moved onto something else. Of course, everyone was dying to know what the single problem was. At the end of the talk, someone managed to pluck up courage to find out. The answer was, "Oh, basically, people did not know what they were trying to do."

12.03pm **Automatic Generation of make Dependencies**
Kim Walden, SYSLAB & ENEA DATA

Abstract

It is standard practice on UNIX to use the **make** program to keep a system of interrelated modules up to date. When text files contain **include** statements referring to additional text files, the task of manually keeping track of all **make** dependencies implied by such references soon becomes unmanageable. Therefore, the dependencies are often generated automatically from extracted **include** statements.

However, the problem of producing the correct set of implied dependencies is a bit more complicated than it may appear at first sight, and all automatic methods we have seen in use over the past years have been erroneous. Errors in the generated **make** dependencies often lead to acceptance of systems with obsolete parts, which may be very difficult to detect, particularly since the number of dependencies rapidly increases to hundreds, or thousands, even for systems of moderate size. Therefore, the problem was analyzed and an algorithm presented in the article "Automatic Generation of Make Dependencies" by Kim Walden, *Software Practice and Experience*, vol. 14, no. 6, pp. 575-585 (June 1984). The algorithm was demonstrated on a simple example, which may be used for testing dependency generating tools.

Since the time of publication, a number of **make** dependency generating programs have been released over USENET, but none of them handles the test example correctly. Obviously, there is a need to draw further attention to the problem, and show that if certain fundamental points raised in the article are not observed, dependency generating tools will indeed produce incorrect results in very common practical situations.

Comment

Basically, don't use **#include** in complex ways if you want **make** to work correctly.

☛ Lunch ☚

2.30pm Session Chair: **Helen Gibbons, EUUG**

Helen showed a picture of Owles Hall and talked a little about the operation of the office which she runs. Afterwards, she claimed to have told her one joke. I will not repeat it, in case she wants to use it again.

2.35pm **TEX Must Eventually Replace nroff/troff**
Timothy Murphy, School of Mathematics, Trinity College, Dublin

Abstract

TEX must eventually replace **nroff/troff** as the standard UNIX text-formatter, for the output of TEX is an order of magnitude superior to that of **troff**. Thus:

(1) TEX reads a whole paragraph before deciding where to break the lines;

(2) In TEX the space between two letters depends on both, in **troff** only on the first;

(3) The spacing around mathematical symbols in TEX depends on the "function" of that symbol: binary operator, relation, left parenthesis, etc;

(4) The size of matching parentheses in a mathematical formula is automatically adjusted in TEX to the height of the expression they enclose.

But if TEX is to be integrated into UNIX it must undergo major surgery.

This paper discusses the advantages of TEX and the problems of properly integrating it into a UNIX environment.

3.11pm **Computer Aids to Rewriting and Copy Editing of English Text**
L. L. Cherry, AT&T Bell Laboratories

Abstract

Writing is generally thought of as a three stage process: planning, producing a first draft, and revising and editing that draft. The UNIX Writer's Workbench† software is a set of programs to help writers of English with the last stage – revising and editing. It includes programs that analyze the writing style of the text, provide the writer with other views of the text that may identify the parts that need revision, and help in the copy editing process. In this paper I will describe the programs, discuss how they have been used, and how they might be extended to help writers who are writing in English as a second language.

Comment

The Writer's Workbench consists of two main programs. The first, **proofr**, is a shell script which runs five other programs: **spellwwb**, an augmented **spell** program with the ability to look in a user's private dictionary; **punct**, a punctuation checker; **diction**, which prints misused and wordy phrases from a dictionary of 450 phrases and in turn runs **suggest** which gives alternatives to the phrases; and finally, **gram**, a rudimentary grammar checker.

The second part of **wwb** is **prose**, which compares the values from a table of statistics computed from the text with a set of standards and produces a two or three page description of the text in terms of a set of stylistic features. This program consists of two other programs: **parts** and **style**.

†Writer's Workbench is a trademark of AT&T Bell Laboratories.

☛ Coffee ☚

Michel Gien managed to arrange for the technicians in the hall to tape the final session on cassette for me – for which many thanks. This means that you can have most of it in full gory detail. Actually, I think that it reads very well.

Teus Hagen started the session by thanking the technicians in the hall, the interpreters, and the local conference organisers, especially Michel Gien and the AFUU. He also thanked Network Events for organising the exhibition.

He introduced David Tilbrook, who was to chair the session. David got a very large ovation because he was wearing a suit, tie, and shoes. He was presented with a Swiss Army knife with 124 tools as a memento of his time in Europe and in recognition of the work which he has done for the EUUG. This replaced his own knife with only seven tools. Teus fooled David into using this miniscule piece of hardware to extract the new upgraded model from the enormous box in which the new knife had been packed. Teus failed to read out what all the 124 tools were.

4.19pm **I2U**
Luigi Cerefolini

The Italian UNIX User group has sensibly compressed the two U's and made something which is much easier to say. A G at the end is optional.

The group was started in July 1984 and has 49 members. They have had several meetings of the Executive Committee, and the result was two meetings in the last months of 1984, including an exhibition. They are preparing a newsletter; number one is in the press. They would like to join EUNET, but they have a problem with the national telephone company, which does not provide any form of auto-dialling modem.

"We are not discouraged by this and we have two projects to solve the problem. The first is to train people to dial numbers during the night while they are asleep. There are some good results with this, but the problem is that people tend to dial the wrong number.

"The other project is to have a dedicated network in Italy. We will use completely autonomous lines and are thinking of a new type of cable. The cable will be made of spaghetti. This is much better than other cables like fibre-optics, TV cable or telephone paths because it is a *de-facto* standard. It is supported by industry, the spaghetti industry, and is available on world-wide basis, which is very important. Last, but not least, the other important thing is that it is edible."

Luigi reminded us that the next Spring meeting of the EUUG will be in Florence.

David then announced, "I have just received a bulletin from the Department of Industry in the United States. All wheat shipments to the Soviet Union are being held back to ensure that there is no technological advantage which the Russians can gain by having spaghetti manufacturing materials."

4.24pm **CUUG & LUUGACAS**
David Tilbrook

David went onto give a couple of "news from" items, they don't translate well onto the printed page because they relied heavily on the use of overhead projector slides.

The CUUG, the Canadian UNIX User Group, was formed instantly at this meeting by David, who elected himself onto the Governing Committee so that we can pay for him to come to EUUG conferences.

The LUUGACAS, which stands for the London UNIX User Group and Curry Appreciation Society, meets every last Thursday in the month (excluding December) in the Lyric Tavern,† Great Windmill Street, London W1. David claimed that this was world's most active UNIX group. The group was formed about 18 months ago, and members from the group figure very heavily in this session because two of the debates were set up there.

†Check on the venue with Jim Oldroyd or Mike Banahan of the Instruction Set; they are thinking of moving it.

4.30pm **The Final Session – Debates**
Various artistes

I think that one of the good conference fixtures which David has instituted is the "final session." He started by explaining the history of this.

"We had panel discussions at the two previous conferences. At Nijmegen, we were blessed with the presence of Larry Crume, Kirk McKusick, Hendrik-Jan Thomassen, Adrian Freed and Mike Banahan. We had Larry Crume representing the System V view; two places away from him was Kirk McKusick; and in the middle was Robert Ragen-Kelly toggling his environment switch. The panel was relatively successful. We also had Andrew Hume and Eric Allman in the audience, which meant that the audience drowned out the panel.

"We tried this again in Cambridge and it wasn't quite so successful. This can best be explained by the fact that there was a question from the audience asking: 'which would the panel choose – 4.2 or System V?' Five people on the panel said 4.2. Tom Killian said V8, at which point, four people changed their minds and said V8 – if we can get it. But it did show that there wasn't enough difference. Mike O'Dell was in the audience, he participated a lot, but it didn't get the spirit which we wanted. There was no blood letting. So, we set up debates for this conference. The debates, I hope, will be amusing and informative. When we suggested this, David Turner said that the debates would be likely to shed more heat than light. I hope so."

There was a slightly formal format where the speaker for the motion was allowed three minutes to state their case; the speaker against the motion spoke for 4 minutes; and this was followed by a one minute rebuttal by the first speaker. Five minutes of discussion from the floor was then allowed. Jim McKie acted as the "Sergeant at Arms," reading out the title of each debate and biographies of the speakers who had not been introduced before. As I did not manage to get hold of the biographical notes for the rest of the speakers at the conference, I will minimise space by omitting them here also (this document is too long already).

1. **Declarative languages must eventually replace imperative languages**
For: David Turner, UKC
Against: Tom Duff, AT&T Bell Laboratories

For: David Turner

"Mr. Chairman, Ladies and Gentlemen – and C hackers. David Tilbrook told me to cut the technical arguments and go straight for personal abuse. I want to move the motion that declarative languages must eventually replace imperative languages, and preferably sooner rather than later. Existing programming languages, the ones in common use anyway, are based on the same basic model of computation. They have the assignment statement as their basic step, they are sequential, and are based on side effects, and so on. This is a model of computation that emerged in the late 50's. Fortran is really the paradigm, or perhaps Fortran and Cobol, and all modern languages can be considered dialects of Fortran.

"One has honest dialects of Fortran such as C, and dishonest dialects of Fortran such as Pascal or Ada – which are pretending to be something else. I submit that there is something fundamentally wrong with this whole model of computation, and something fundamentally wrong in the way that we produce software at the moment.

"There are two main symptoms of this. First of all, software is too expensive to produce, and secondly, once produced, it is usually wrong. It usually has errors, or "bugs," as we call them.

"Let me take the correctness problem first. Most non-trivial programs have errors; that's a situation which we have gotten used to. Many large programs upon which we depend from day to day are riddled with errors; this is really a terrible situation that we ought not to be willing to put up with. We try to eliminate errors after the event by a process called "debugging." Now, I want you to compare that with the way which other engineering disciplines proceed. It is not the case that they design aeroplanes by debugging them. They don't put an aeroplane up in the sky, watch it crash in a

nose-dive, and say 'Oh dear, I think that the wings must have been the wrong shape; let's try again.' They have mathematical models of what makes aeroplanes fly, and they do a lot of calculations first. They know to within a very fine degree of accuracy how that aeroplane will behave before they build it. We need to do the same with software. We need to have proper mathematical basis for programming.

"I submit that no language containing an assignment command can have reasonable inference properties. Declarative languages do have reasonable inference properties. They are also much more compact. So, if we switched to declarative languages, not only would we have a proper formal basis to make possible formal development of programs and formal verification of programs, but also programs would be much shorter and much quicker to produce. Finally, because they are not sequential and not based on side effects, they have the capacity to take advantage of new kinds of very highly parallel hardware that VLSI makes possible. I urge you to support the motion."

Against: Tom Duff

"Arthur C. Clarke once said that if a scientist tells you that something is true then he's probably right, if he tells you something is impossible he is very likely to be wrong. I feel like that I am in a bad position on that ground. But, there are a number of things to say.

"Niklaus Wirth (or Nicholas Worth as we call him in America) once said if housebuilders built houses like computer programmers build programs then the first woodpecker that had come along would have destroyed civilisation. Which is roughly a paraphrase of one of David's points. Wirth obviously never saw a carpenter build a house.

"Secondly, I have a friend who worked at Boeing for many years, and he said that the average Boeing 707 has something like three tons of shims in it. So, the argument from other engineering methods doesn't hold up at all.

"As for formal verification, it is fine for mathematicians and fine for people who are working on *very* small problems. There exist no significant systems that have ever been verified, done in any sort of language, let alone the sort of nonsense that my worthy opponent advocates. It requires Ph.D mathematicians to understand this sort of stuff; I got out of mathematics at an early age because it's just too tough for my feeble brain. The general class of people that you have writing computer programs are not up to it. I include myself; but, of course, not the audience.

"The fact that formal verification methods cannot handle languages with assignment is not the fault of the languages, it's the fault of the methods.

"Languages without assignment don't match current hardware very well at all, and if you think that you are going to change that situation, then I think that you're dreaming. We have had Von Neumann machines, machines with big memories and little CPU's, since before I was born, and I think that the chance of that situation changing at all is negligible at best.

"There are plenty of problems where the assignment statement is precisely the right model to think about the problems. For example, a case from interactive computer graphics: take a frame buffer, which is just a large memory, and say you want to draw pictures in it using a mouse or a digitising tablet. The model you have there is taking something that's shaped like a paintbrush and plopping it down in the frame buffer. It's precisely an assignment model which describes that sort of problem. There doesn't seem to be a good way of describing it otherwise.

"The applicative models don't describe important features of a lot of algorithms. An example which came up the other day was quicksort, where there are two important properties to the algorithm. One is the notion of recursive sub-division, which the applicative model handles just fine; the second is the notion that you can do the whole thing in place, and there is no place for that sort of idea in applicative languages.

"A text editor is something that you are going to have a lot of trouble describing if you have no notion of a variable. Where are you going to write the file out? The idea is that you read in the file, you make some changes, and you *put it back where it came from.* The idea just doesn't exist in the applicative model.

"Portability is another issue."

Tom was cut off by time but managed to slip in, "that's about all I have to say. And besides, Turner is a foreigner and can't possibly know what he is talking about."

Rebuttal: David Turner

"I can't possibly pick up all of the points. But first, let me say that of course we don't know how to program everything in the functional style. As it happens, we do have a working interactive text editor. There's obviously no time here to tell you how it works, but you can solve that problem. There are other problems that we don't yet know how to solve, but that, of course, is why functional programming is fun. It's a frontier; there are still new things to find out.†

"Formal verification doesn't work, we are told, because it only works on toy examples. Two points about that: of course it doesn't work at the moment because we are using languages with the wrong fundamental properties. If you switch to languages with the right properties, like functional languages, formal verification becomes very much easier. Of course, like everything else, formal verification hugely benefits from the intervention of computers. I don't advocate that people do fully formal verifications by hand; that's crazy. I advocate that we build computer systems to do verifications for us, computer systems steered by a human being. I believe that is possible on the basis of functional languages.

"If it's the case that the hardware we have now doesn't match functional languages, let's build new hardware."

Comments from the floor:

Mark (Seiden?)

"I'd like both speakers to comment on the ease of debugging large programs written using their respective styles."

Tom Duff

"There are no programs written in Turner's style. He cannot debug them."

David Turner

"Not true, not true. There are no large programs; that's because programs become brilliantly small when written in my style."

Tom Duff

"They'll get small in APL too; it's got nothing to do with functional style, it's got to do with big operators."

David Tilbrook

"Fault; it's fifteen love."

Eric Allman

"I can't help but notice that in your functional language, you've hidden that fact that you actually do have state by putting things into huge vectors. You then run through them and essentially pretend that what you really have is all the values of the variable stuffed into one, in a convenient little way. It seems to me that there still is state in your language."

David Turner

"Yes, I think that's a legitimate comment. What happens is that instead of having a sequence of states in time, we represent it as a sequence of values in a data structure; so it looks like space instead of time. That is exactly the trick which the physicists use to describe the world. That's exactly what Newton did when he invented the equations that we use to describe dynamics: you treat time as a dimension of space. That's really what functional programming is advocating: that you handle state change by having a data structure that represents the successive states, and the whole

†Tom Duff: "That wasn't the argument."

thing can be conceived of statically instead of dynamically and that makes it more amenable to reasoning. But your comment is legitimate."

Comment

David Tilbrook did say at this point that debaters do not necessarily hold the views that they are stating. Your roving investigative journalist can now reveal that Tom and David did have some difficulty in finding something to disagree about.

Ah well, onto the next debate.

2. **Version 6 was the last real UNIX**
For: Dave Lukes, The Instruction Set
Against: Mark Seiden, Lucasfilm & IBM

Dave was dragooned into standing in Mike Banahan's socks when Mike was unexpectedly called back to London.

For: Dave Lukes

"What's good about V6? Those of you who remember V6 will remember an amazing thing, which is the shell. The shell did not have nice structured programming constructs; it had **goto**. The good thing about **goto** was that it was a separate program. People are talking about unbundling UNIX nowadays. V6 was great; if you didn't like having a high-powered shell, you removed the **goto** program.

"People argue about what are the bad points with the things which came after V6. Here's my favourite bad point: the program line

```
wc -b9600
```

What's this number 9600? It's something to do with baud rates. That's the kind of thing that happens when you start extending a nice clean simple system. That was 4.1BSD; there are worse things that have happened since. I won't say that 4.1BSD is the worst offender.

"V6 **pr**: look at the **pr** program and count the number of flags. Most people give more flags than filenames to **pr**, all the time. There's something wrong there. It's got worse: **pr** has had more flags with every single release.

"The shell, remember that? A nice simple little thing. You give it a file, maybe, and some arguments. Now, the V7 shell has the **-ceiknrstuvx** flags. What do these arguments do? Well, they're all really interesting, you know. Like one of them does things like, I think **v** prints out its input. Ever heard of **tee**? There are a lot of useless flags in the world. I don't know what the rest of them do, but I am sure that they're quite interesting as well.

"V6? It's simple, it's cheap, it's extremely nasty but it works. The shell is 20 pages of C in Version 6. Beat that anybody else. What else can you say?

"That's V6, the greatest UNIX that ever lived, simply because you could carry it around in your briefcase."

Against: Mark Seiden

Mark's talk was punctuated by pictures from the history of transport. He started with early stationary engines, passing through vintage cars, steam trains, and ending with British Rail's Advance Passenger Train. Naturally, the pictures cannot translate to this document so a certain amount of imagination is required, but we sat and laughed a lot at the pictures.

"The proposition is 'Was Version 6 the last real UNIX.' I think that if you look at it carefully, you'll realise that it wasn't a very real UNIX to begin with, and even if it was real, other later systems are much more real. But, first some history to help put us in focus.

"In 1969, Ken Thompson first wrote the game of Space Travel for a little-used PDP-7 at Bell Labs in Murray Hill. Within the year, due to Ken and Dennis Ritchie, the first UNIX system was

written in PDP-7 assembler. Even by then, it was an obsolete machine. The first version for PDP-11 was in assembler as well.

"Some four years later, the system had evolved considerably to Version 6. It was now capable of real work, although it did have many limitations. Sizable pieces of code were still in assembler, and the code still ran only on PDP-11's. It couldn't support heavy peripherals, mass storage devices, because of 16-bit block addresses. The maximum size of a file system was 32 mega-bytes. It made no claim to satisfy real time requirements or single user requirements. It certainly didn't satisfy the needs of the European UNIX community, so people started hacking it in a heavy way.

"So, let me ask you. Could you do now without the Bourne shell, the back-quote operator, **pcc**, **lint**, **make**? They weren't in Version 6. So, Version 7 was the first real UNIX. It had those things and more importantly, it was portable. What does portability mean? It means independence from a single manufacturer and can be translated directly into personal freedom for UNIX programmers. It means you can work almost anywhere.

"One problem with Version 7 was the file system. A power failure, or a test kernel scribbling all over the system buffers could cause great unhappiness. If you remember **icheck**, **dcheck**, and **ncheck**, and trying to apply them in the right order at two in the morning, you'll know what Version 7 was like. **fsck** came along just after Version 7.

"The system still didn't support virtual memory, and you needed a workhorse like Berkeley UNIX to do that. Berkeley UNIX versions added support for virtual memory, a full screen editor, **vi**, auto-configuration, and then later, flexible usable networking. A faster file system on 4.2. Can you do without **fsck**, history, job control, symbolic links, Ethernet? Perhaps.

"Perhaps we should be asking: is UNIX real, even today. It still needs **fsck** every time you boot, you still have to get out of the mail reading program when new mail has arrived, that is ridiculous. The world has changed in 10 years, and reality has to reflect those changes. Now, ordinary people use computers every day. We have semi-conductor memories, 5 (and a quarter) inch discs with 500 megabytes on them, optical discs soon to come, and wide choice of networking technologies. There may not be a real UNIX even today, I think. Perhaps in fact, the last real UNIX might be **Multics**."

Rebuttal: Dave Lukes

"Well, I'll knock these things down one by one.

"First, he says that V6 is not portable. I used to carry it around everywhere. I had a listing in my briefcase, and what's more I could get a sandwich and an apple in there with it – and the manuals. I admit that I didn't eat much lunch; Tilbrook had a bigger briefcase.

"Next assertion, everything is too small the guy says. 2^{24} bytes? When was the last time you made a 2^{24} byte file? Next, 2^{31}. Is that big enough? No way; discs are bigger than that now. Also, if you have a 2^{31} byte file, there is no standard UNIX utility which can manipulate that object. You can purchase databases which can do it, but there is no standard UNIX utility which can do anything with a file that big. So, what do you need it for?

"They say it doesn't have **awk**. Well, people don't really miss **awk** because all they ever get out of it is 'p&P6'. That's what lives at location zero on the PDP-11: it's the bit of code that you get from printing out a null pointer.

Comments from the floor and other diversions:

Mark Seiden

"Dave, you are wrong; there is a utility which will operate on a file that large, and you mentioned it at the beginning of your talk – **wc** – at least on a VAX."

Eric Allman

"**rm**?"

David Tilbrook

"Eric Allman, the master of small programs..."

Eric Allman

"I'll get you for that.

"Now, you see, you're wrong. The problem with Version 7 is not the code itself but the documentation. With 6250 tape drives you could still put the system into your briefcase. The problem is that they documented everything in Version 7 which they hadn't in Version 6. If you wanted a different option out of the program, then you went into the program and changed it, if the option that you wanted wasn't already there."

Unknown speaker

"The biggest mistake with the later UNIX versions which was not mentioned was that it opened up room for lousy PC-DOS. Right now, there is no UNIX system around which is really small enough and uncomplicated enough that it can stand up in the very low end of the personal computer market."

Mark Seiden

"But the reason for that, it seems to me, is a marketing one. That is the enormous market penetration of IBM. Application developers choose to write their application to run in the PC-DOS environment. But, to many of their credit, they write them in C so they will also run in the UNIX environment if they see a business plan for it; for example, **multi-plan**. I don't think that there is a technical argument."

Radek Linhart

"I have heard from somebody that there is a UNIX which fits into ROM's, and you can carry it in your briefcase."

Mark Seiden

"I have here an optical disc on which is 4.2, all its documentation, and all historical versions of UNIX before that. It fits very nicely into my briefcase."

3. **vi is a piece of wombat do**
   For: Nigel Martin, The Instruction Set
   Against: Tim Snape, Chameleon Software

I'd like to insert a little from Nigel's biography because it contained a great line: "He was promoted from a lecturer in Computer Systems at University College, London, to a junior operator as a result of using UNIX."

For: Nigel Martin

"Let us begin at the beginning of time when we all got used to using **ed**. We all enjoyed **ed**, and it had a number of facilities. One of the biggest downfalls of **ed** was that you had a lot of confusion associated with using it, and you wanted to be able to view larger amounts of your file. Therefore, along came an editor called **vi**. Its name: "visual." Everyone was extremely pleased about the arrival of **vi**. Hopefully, as the name implied, "visual" would allow you to see what was going on with your edits and allow you to see a substantial portion of your file. Unfortunately, this appears not to be the case. **vi** shows you a picture and allows you to see a number of characters. Regrettably though, there is an extensive amount of user confusion still present.

"Invariably, you will sit at your keyboard and you will type a number of characters. Those characters will sometimes, if you are very fortunate, have an effect. They may, if you're even luckier, be inserted into the file at exactly the right place which you wanted. This experience is extremely rare.

"It is far more likely that when typing in your characters, you will type a number. For instance, '1'. Nothing happens; you look distressed. Is it because the machine is slow? Is it because **vi** is

slow? No. It's because you have just typed a '1' at the wrong moment. So, you type a '2'. What happens? Nothing. You are equally distressed; you look around in absolute bewilderment. So you type another number: '3'. Still nothing happens. You realise that **vi** clearly does not like numbers, so you try and go for some letters. You hit a 'd'. What happens? Nothing. You then hit a second 'd' and hey presto! The picture changes – 123 lines disappear from your file!

"It is argued that the facilities of **vi** allowing you to prefix a command with a number are extremely important. It is unfortunate, though, that this is not extended to every command; consider, for example, the tilde command.

"One of the rules of UNIX is that UNIX is built out of a number of tools. Therefore, we would have hoped that an important and useful UNIX tool was the screen editor. We all know that one of the most fundamental rules of a tool is that: silence is golden. If you walk into a room of people using **vi**, you will realise that **vi** clearly is not a tool. There is an extensive amount of sound. The sound comes from people screaming, from **vi** ringing the bell, and from many, many, other sources. It is evident, therefore, that **vi** cannot possibly be considered to be a tool.

"Finally, in order to make a very very trivial edit to your file, perhaps as a result of a small typing mistake, you will have to type numerous keys to shift you in and out of command and edit mode. This is most unfortunate. Clearly, if you want to move the cursor up, the same key should allow you to move the cursor up at all times. You should not have to go through the process of negotiating your way around this supposedly useable tool. I therefore urge you to consider that **vi** is not only unsuitable, that it certainly is not a tool, and it certainly is not a screen editor."

Against: Tim Snape

"I don't know if you know this, but in fact, 'wombat do' is very high in nitrates. Certain small South American countries have based their economies on such products, so IBM take note.

"I would like to say that I use **vi**, and I like it. I will itemise the points which I find attractive.

"The most important benefit of **vi** is that it is standard. You can go from one UNIX machine to another, and you will still have the same screen editor on it. You will be able to edit your files using a screen editor.

"Another benefit is that it uses *termcap*. This means that you can customise **vi** to work on even more bizarre and weird terminals as they become available.

"The heritage of **vi**: Nigel briefly mentioned that **ed**, well, in fact, **vi** is based to a small or lesser extent on **ed**, **sed** and **ex**. It's a family of text editors. While I was researching this talk, I tried to discover the parentage of **vi**; it was rather difficult.

"What it means is that once you have trained your first-year undergraduates to use **ed**, something which can be done fairly rapidly, they can move up and use **vi** in a fairly easy way. (Fingers crossed.)

"The lack of *wysiwyg*-ness, the problems that Nigel has been having with **vi**, are in fact not bugs at all, but features. What **vi** is attempting to do is to minimise the amount of character I/O which is taking place over your terminal line. So, you can use **vi** over PAD's and 1200 baud lines. You cannot do this with some of the other visual editors.

"**vi** has a lot of functionality, a lot of power. It's been said that real programmers are not really worried about the cosmetics, they are more concerned with power. Multi-modal editors do offer a lot of power. One of the things which **vi** has, which I like, is the syntax-directed extensions: the ability to tab around functions and the ability to tab around curly brackets. I find that very useful. Also, the undo facility and the ability to pass text through UNIX filters are both very nice features.

"That's not the point; the real point that we are discussing today is: should we be using single mode editors or should we be using multi-mode editors? Single mode editors are attractive to secretaries, typists, English students, and people like that. Multi-mode editors are very attractive to people who like to have a powerful command structure. It has been said that multi-mode editors increase the thinking time for users. I think that this is probably a good thing for some C programmers.

"**vi**, like UNIX, like C, has been criticised for its lack of user friendliness, it has been criticised for its complexity; but these aren't real arguments. Like C and UNIX, **vi** is a tool which grows on you, the more you see other people using it, the more you like it and the more you learn.

"Finally, Nigel, if you do have problems using **vi**, perhaps you would like me to run a course for you."

Rebuttal: Nigel Martin

"In the introduction to Mr. Snape here, my learnèd collegue on the other side, he informed us that he was interested in producing tools and utilities to aid programmer productivity. It is unfortunate that the very tool or thing he describes does not quite come up to this.

"**vi** is standard. Yes, unfortunately it is. We know there are many things wrong with many standards. How many people here are happy with any standards they know of?

"With regard to the use of *termcap*: perhaps, Tim, you would like me to supply you with a free copy of a *termcap*-driven **ed**.

"As regards the family information, it is most amusing that you are unable to trace the parentage of **vi**; perhaps that is something to do with its quality.

"With respect to character I/O: I would suggest to you, ladies and gentlemen, that **vi**'s primary objective in life is to try to melt any form of communications facility. Every character you type results in you having to use the undo command to put back all the mistakes you have just made.

"Finally, I am sure that many programmers invariably think that they are undergoing a secretarial task. Therefore they would like to relax, concentrate on their program, and not have to pay too much attention to trying to negotiate their way around these things which are obstructing us in our way. The one thing that UNIX was trying to do when it was originally designed in '69 was to allow people to get on with their job, that of programming, rather than having to fight with the system. So, I would urge you that **vi** is not an appropriate tool to do this."

Comments from the floor:

Erik Fair

"Mr. Martin, I am curious, since you are a detractor of **vi**; which editor do you prefer to use?"

Nigel Martin

"It is most unfortunate; **that** is not the subject of this debate."

Eric Fair

"Oh dear, that doesn't help at all. How can I detract from your personal heritage if I don't know what editor you use?"

Nigel Martin

"I am sure, Sir, that this is of no importance to the discussion in hand."

John Haxby

"I'd quite like to know why **vi** was written in the first place. We had a perfectly good multimode editor with none of the features of **vi** long before **vi** even appeared in the UK."

Nigel Martin

"Mr. Chairman, I object to the word 'feature' being used in the same sentence as **vi** without a *not*."

Dave Rosenthal

"It's a while since I used **vi**, but I've just heard a load of people saying things about the undo command. As somebody who uses an editor with a real undo command, like **emacs**. There isn't an undo command in **vi** because if you undo it, it undoes itself."

Comment

There were some more comments in this vein; nobody was really brave enough to stand up and say anything much in favour. Eric Allman did make a slight stand, but I don't think that he scored many points! I guess **vi** lost here.

Well, there were only three debates in the official order of events, and the next one was a surprise to us all. Jim McKie (wearing what might be described as "hot pants" – or am I showing my age) read out the resolution.

4. **Jim McKie has nice legs**
For: Jean Wood, DEC
Against: Louise Sommers, IST

Both speakers in this debate used some wonderful slides to complement their talks. The slides showed giraffes, rhinos, hippos, chimpanzees and other fauna. Oh, the first slide was a picture of Jim showing the main topic for the debate – his legs – "where did you get that?" asked Jim.

For: Jean Wood

"We have to consider this from a serious point of view. Both functionally and aesthetically, those are great legs. Of course, their main function is to connect the top half of the body with the ground, and you can see they do that exceptionally well. They're straight. They're strong. They have knees in the right places. I guess in the language of computerese we could call those 'very robust legs.' On the other hand, they're also very user friendly. They have the correct quantity and quality of hair. The knees have kneecap worthiness.

"It's very difficult for me to look at this and speak unemotionally. To put things into perspective, I thought that we could look at some similarities from nature and compare them with Jim's legs." *Here were the various beasts.*

"To be honest, I have had very little opportunity to look at Jim's legs, except at conferences like the rest of you. I talked with an expert, his wife; she's a nurse and she spends lots of time doing things like bed-bathing men, and she has seen lots of legs. Her comments were: 'Jim's legs are nice and straight. With a nice pair of legs, the less attractive appendages, like chins, don't matter'."

Against: Louise Sommers

"I am speaking against the motion. My honourable opponent has described Jim's legs in glowing terms. But I feel that she has picked the wrong comparisons." *More pictures of fauna.*

"My opponent also quoted Gilly McKie in saying that 'Jim's legs were nice and straight,' but she didn't give you the whole quote. When do they ever? What Gilly actually said, and I wrote it down at the time, was that 'Jim has nice straight legs' (she said this in the most beautiful Scottish accent, which I can't do) 'but they're not very muscley, you know. If he's going to wear a kilt, he'd have to wear very thick socks as well.'

"Being a expert on men's legs (and bottoms I might add), I did a little survey while I was here at the conference and I looked at all the men's legs. I want to tell you what nice legs really are, and they are David Tilbrook's. Would you please show your legs, David."

At which point, Mr. Universe stood up, ripped off his very smart trousers, and demonstrated (or perhaps one should say "flaunted") his legs. Of course, Louise is somewhat biased since the large Canadian and Louise are married. She went onto say, "Now, these are nice legs, well shaped, firm, strong, and believe me when I say *sexy*."

There was one further comment from the floor, an unknown speaker.

"I've heard that Jim's legs are bug-ridden, is that true?"

Jim McKie

"No; my legs are written in a functional programming language."

☛ End of day 3 ☚

The evening event was a concert of electronic music at IRCAM. I must confess to have decided to miss this event, largely on the grounds that I **know** that I don't like that sort of stuff. I asked Jean Wood to write me a short piece on the concert, and hopefully she will.

The exhibition was open the next day, and I spent most of the time there, apart from about three hours sitting outside a café in the sun; isn't Paris wonderful? Anyway, I digress. I suppose that the most noticeable thing at the exhibition was the emergence of the colour display; there seemed to be a colour display on almost every stand.

The other thing which caught my eye was really nothing to do with UNIX, it was the French PTT terminal which they are putting into every home and office in certain parts of Paris. This gives access to an electronic phone directory, electronic mail and telex facilities. My brother has one of these in his office and was enthusing about what he could do on it.

I spent a lot of time playing with a colour Sun workstation on the Gould stand; this is fun, but of course pricy.

Endpiece

Well, another conference gone. We will be learning some lessons from this one, just as we have learned from previous conferences. I have always thought that every conference is an experiment in doing conferences. I traditionally give private awards in these reports. We didn't have a silly competition at this conference, there was no publically available blackboard to write things on, and we couldn't really come up with something silly enough. So, the private awards.

The "best freebee from an exhibition stand" award goes to the Instruction Set for their wine labelled "Chateau NUXI," which is the name of their UNIX system (not the Chateau, the NUXI). I took the bottle back to my brother's flat with the thought that he could always use it for cooking if it was vile – but it wasn't, we drank it. I suppose that you could say that I have no wine discrimination facility.

The "bore of the conference" award must go to the French customs official who confiscated all the EUUG T-shirts which had not already been given out because they were made in Turkey. We could get no convincing explanation of why Turkey's T-shirts should cause such offence.

The "event of the conference" award goes to a deserving failure. An attempt was made on the world record for mismatching shoes. The record, previously held by the British Army in the Crimea (who took delivery of several hundred left boots), has stood for more than a hundred years. The attempt involved getting as many people as possible in one place wearing mis-matched footwear. It's unfortunate that there were just not enough people involved in this event.

The "coming out of the closet" award goes to Nigel Martin, who managed with great difficulty to disclose in a suitable small circle of close friends that "he is a **vi** user." I always thought that Nigel was a bit that way inclined; perhaps it is the way he walks.

The (more serious) award for the fixer for the conference should go to Helen Gibbons. She dealt with a great many problems in an uncomplaining way. Not many people were aware of the problems at the time, because by the time conference attendees had gotten there, the problems had simply gone away.

Oh well, I must stop. This document breaks size records. Thanks to all who made the conference work.

USENIX Association Financial Statements

November 30, 1984

USENIX Association

The accompanying balance sheet of the USENIX Association as of November 30, 1984, and the related statements of revenue and expenses and changes in financial position for the year then ended have been compiled by me.

A compilation is limited to presenting in the form of financial statements information that is the representation of management. I have not audited or reviewed the accompanying financial statements and, accordingly, do not express an opinion or any other form of assurance on them.

Elliott D. Buchdruker
Certified Public Accountant

Pier 15, The Embarcadero
San Francisco, CA 94111

USENIX ASSOCIATION
BALANCE SHEET
NOVEMBER 30, 1984
(Unaudited)

ASSETS

CURRENT ASSETS:		
Cash in bank:		
Mechanics Bank	$ (186)	
First Interstate Bank	108,205	$108,019
Marketable securities		185,726
Prepaid conference costs		42,212
Prepaid expenses		16,869
Prepaid supplies		6,332
Total current assets		359,158
FIXED ASSETS, AT COST	18,465	
Less – accumulated depreciation	(4,993)	13,472
		$372,630

LIABILITIES AND FUND BALANCE

CURRENT LIABILITIES:		
Accrued expenses		$ 12,731
Total current liabilities		12,731
FUND BALANCE:		
Balance, December 1, 1983	$143,314	
Excess of revenue over expenses for the year ended November 30, 1984	216,585	359,899
		$372,630

See accompanying notes to financial statements.

USENIX ASSOCIATION
STATEMENT OF REVENUE AND EXPENSES
FOR THE YEAR ENDED NOVEMBER 30, 1984
(Unaudited)

REVENUE:	
Conferences and Workshops – net income (Exhibit A)	$291,162
Membership Dues	128,178
Publications	17,971
Manuals	15,428
Dividends on Investments	7,518
Miscellaneous	1,425
Interest on Bank Accounts	252
	$461,934
EXPENSES:	
Salaries and Benefits	$47,445
Meetings and Travel	18,303
Conferences	16,963
Tape Services	16,955
Telephone	16,132
Printing and Duplicating	15,603
Postage	14,088
Office Management Fees	14,000
Newsletter	13,718
Liquidated Damages	12,000
Computer	8,986
Rent	7,091
Legal and Accounting	6,939
Outside Services	6,353
Repair and Maintenance	5,354
Usenet Software Development Project	4,115
Office Supplies	4,039
Depreciation	3,477
Consulting	2,973
Equipment Rental	2,226
Freight	2,059
Refunds	2,058
Insurance	1,500
Taxes and Licenses	1,178
Penalties	639
List Maintenance	524
Bad Debts	420
Miscellaneous	211
	245,349
REVENUE OVER EXPENSES	$216,585

See accompanying notes to financial statements.

USENIX ASSOCIATION
EXHIBIT A
CONFERENCE AND WORKSHOP INCOME
FOR THE YEAR ENDED NOVEMBER 30,1984
(Unaudited)

	Revenue	Expenses	Net
Washington, D.C. Winter Conference	$174,660	$14,611	$160,049*
Salt Lake City, UT Summer Conference	331,751	204,061	127,690
Newport, RI Workshop	12,775	9,352	3,423
TOTALS	$519,186	$228,024	$291,162

*The Washington D.C. conference was the first UniForum Conference jointly sponsored by the USENIX Association and /usr/group. General and joint expenses were paid before $174,660 was disbursed to each of the two sponsors. The $14,611 expenses were for strictly USENIX Association activities.

See accompanying notes to financial statements.

USENIX ASSOCIATION
STATEMENT OF CHANGES IN FINANCIAL POSITION
FOR THE YEAR ENDING NOVEMBER 30, 1984
(Unaudited)

SOURCES OF FUNDS:	
Net income	$210,253
Add – depreciation not requiring a current outlay of cash	3,477
	$213,730
APPLICATION OF FUNDS:	
Purchase of fixed assets	$ 8,358
Increase in working capital	205,372
	$213,730
INCREASE (DECREASE) IN WORKING CAPITAL COMPONENTS:	
Cash	$ 90,834
Marketable securities	57,567
Prepaid conference costs	42,212
Accounts receivable	(3,092)
Prepaid expenses	8,061
Accrued expenses	(12,731)
Income taxes payable	22,521
	$205,372

See accompanying notes to financial statements.

USENIX ASSOCIATION
NOTES TO FINANCIAL STATEMENTS
NOVEMBER 30, 1984
(Unaudited)

1. **Corporate Background**
   USENIX Association was incorporated in 1980. The principal purpose of the organization is to provide educational benefits, including the exchange and communication of research and technological ideas pertaining to UNIX[tm] and UNIX related computer systems. It is a nonprofit public charity established under Section 501 (c) (3) of the Internal Revenue Code. Contributions to the organization are deductible by the donors.
2. **Summary of Significant Accounting Policies**
   Marketable securities are recorded at cost. USENIX Association uses the direct charge-off method of recognizing doubtful accounts. Property and equipment are recorded at cost. The Company follows the practice of capitalizing all expenditures for property and equipment in excess of $200. Depreciation is calculated according to the accelerated cost recovery system for property placed in service after 1980. The lives are comparable to those used under generally accepted accounting principles. For all other property, depreciation is calculated over the estimated useful lives of the assets on both the straight line and the accelerated basis. Conference fees are recorded net of expenses.

Local User Groups

The USENIX Association will support local user groups in the United States and Canada in the following ways:

- Assisting the formation of a local user group by doing an initial mailing for the group. This mailing may consist of a list supplied by the group, or may be derived from the USENIX membership list for the geographical area involved. At least one member of the organizing group must be a current member of the USENIX Association. Membership in the group must be open to the public.
- Publishing information on local user groups in *;login:* giving the name, address, phone number, net address, time and location of meetings, etc. Announcements of special events are welcome; send them to the editor at the USENIX office.

Please contact the USENIX office if you need assistance in either of the above matters. Our current list of local groups follows.

In the **Boulder** Colorado area a group meets about every two months at different sites for informal discussions.

Front Range Users Group
N.B.I., Inc.
P.O. Box 9001
Boulder, CO 80301

Steve Gaede (303) 444-5710
hao!nbires!gaede

Dallas/Fort Worth UNIX User's Group
Advanced Computer Seminars
2915 L.B.J. Freeway, Suite 161
Dallas, TX 75234

Irv Wardlow (214) 484-UNIX

continued...

In the **Washington, D.C.,** area there is an umbrella organization called Capitol Shell. It consists of commercial, government, educational, and individual UNIX enthusiasts. For information and a newsletter write:

Capitol Shell
8375 Leesburg Pike, #277
Vienna, Virginia 22180

seismo!{rlgvax,umcp-cs,andi}!cal-uni!capish

In the **New York City** area there is a non-profit organization for users and vendors of products and services for UNIX systems.

Unigroup of New York
G.P.O. Box 1931
New York, NY 10116

In **Minnesota** a group meets on the first Wednesday of each month. For information contact:

UNIX Users of Minnesota
Carolyn Downey (612) 934-1199

In the **Atlanta** area there is a group for people with interest in UNIX or UNIX-like systems:

Atlanta UNIX Users Group
P.O. Box 12241
Atlanta, GA 30355-2241

Marc Merlin (404) 255-2848
Mark Landry (404) 874 6037

In the **Seattle** area there is a group with over 150 members, a monthly newsletter and meetings the fourth Tuesday of each month.

Seattle / UNIX Group
P.O. 58852
Seattle, WA 98188

Irene Pasternack (206) FOR-UNIX
uw-beaver!tikal!ssc!slug

In the **St. Louis** area:

St. Louis UNIX Users Group
Plus Five Computer Services
765 Westwood, 10A
Clayton, MO 63105

Eric Kiebler (314) 725-9492
ihnp4!plus5!sluug

In the **northern New England** area is a group that meets monthly at different sites. Contact one of the following for information:

Emily Bryant (603) 646-2999
Kiewit Computation Center
Dartmouth College
Hanover, NH 03755

decvax!dartvax!emilyb

David Marston (603) 883-3556
Daniel Webster College
University Drive
Nashua, NH 03063

A UNIX/C language users group has been formed in **Tulsa**. For current information on meetings, etc. contact:

Pete Rourke
$USR
7340 East 25th Place
Tulsa, OK 74129

USENIX Association
P.O. Box 7
El Cerrito, CA 94530

First Class Mail

FIRST CLASS MAIL
U.S. POSTAGE PAID
El Cerrito, CA 94530
Permit No. 87

Graphics Workshop

Winter '86 Conference

Domain Addressing in ACSnet

fsck Explained

EUUG Meeting Report

85.1 Software Distribution Tapes

Change of Address Form

Please fill out and send the following form through the U.S. mail to the Association Office at the address above.

Name: _____ Member #: _____

OLD: _____ NEW: _____

_____ _____

_____ _____

_____ _____

Phone: _____

uucp: {decvax,ucbvax}! _____